Ontology-Driven Semantic Annotations for Multiple Engineering Viewpoints in Computer Aided Design

Li, Chun

Award date:
2012

Awarding institution:
University of Bath

Ontology-Driven Semantic Annotations for Multiple Engineering Viewpoints in Computer Aided Design

Chunlei Li

A thesis submitted for the degree of Doctor of Philosophy

University of Bath

Department of Mechanical Engineering

16 May 2012

Table of Contents

List of Figures

List of Tables

Acknowledgments

The author would like to thank his supervisors Professor Chris McMahon and Dr. Linda Newnes for their support and guidance through the study of these years. And also he would also like to thank all his colleagues in the department who have always been happy to help.

He also thanks his parents Hanwen Li and Ruiqing Zhang, his wife Hejia Wang, and the whole family who have supported him constantly in many ways.

The work reported in this report has been undertaken as part of the EPSRC Innovative Design and Manufacturing Research Centre at the University of Bath (grant reference EP/E00184X/1). The authors gratefully acknowledge this support and express his thanks for the advice and support of all concerned.

Abstract

Engineering design involves a series of activities to handle data, including capturing and storing data, retrieval and manipulation of data. This also applies throughout the entire product lifecycle (PLC). Unfortunately, a closed loop of knowledge and information management system has not been implemented for the PLC. As part of product lifecycle management (PLM) approaches, computer-aided design (CAD) systems are extensively used from embodiment and detail design stages in mechanical engineering. However, current CAD systems lack the ability to handle semantically-rich information, thus to represent, manage and use knowledge among multidisciplinary engineers, and to integrate various tools/services with distributed data and knowledge.

To address these challenges, a general-purpose semantic annotation approach based on CAD systems in the mechanical engineering domain is proposed, which contributes to knowledge management and reuse, data interoperability and tool integration. In present-day PLM systems, annotation approaches are currently embedded in software applications and use diverse data and anchor representations, making them static, inflexible and difficult to incorporate with external systems. This research will argue that it is possible to take a generalised approach to annotation with formal annotation content structures and anchoring mechanisms described using general-purpose ontologies. In this way viewpoint-oriented annotation may readily be captured, represented and incorporated into PLM systems together with existing annotations in a common framework, and the knowledge collected or generated from multiple engineering viewpoints may be reasoned with to derive additional knowledge to enable downstream processes. Therefore, knowledge can be propagated and evolved through the PLC. Within this framework, a knowledge modelling methodology has also been proposed for developing knowledge models in various situations.

In addition, a prototype system has been designed and developed in order to evaluate the core contributions of this proposed concept. According to an evaluation plan, cost estimation and finite element analysis as case studies have been used to validate the usefulness, feasibility and generality of the proposed framework. Discussion has been carried out based on this evaluation. As a conclusion, the presented research work has met the original aim and objectives, and can be improved further. At the end, some research directions have been suggested.

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
3DAF	3D Annotation Framework
3DSEAM	3D semantics annotation model
4D	Four-dimensional
AAI	Annotation anchor identifier
Abox	Assertion box
AI	Artificial intelligence
AO	Application ontology
AP	Application protocol
API	Application programming interface
ARC	Areas of relevance and contribution
ASME	American society of mechanical engineers
AW	Application watchdog
BOM	Bill of materials
BPNN	Back-propagation neural-network
B-rep	Boundary representation method
CAD	Computer-aided design
CAE	Computer-aided engineering
CAM	Computer-aided manufacturing
CAPP	Computer-aided process planning
CBS	Cost breakdown structure
CE	Cost estimation
CER	Cost estimation relationship
CNM	Customer needs management
CPD	Collaborative product design
CSG or C-rep	Constructive solid geometry method
CUP	Conceptual understanding and prototyping
DBMS	Database management system
DDD	Document-driven design
DL	Description logic
DMS	Direct material sourcing
DRM	Design research methodology
DSS	Decision support system
DWG	Drawing
DXF	Drawing exchange format
ECAD	Electronic computer-aided design
EKP	Effective knowledge processes
EO	Evaluation outline
EQ	Experimental questions
EV	Engineering viewpoint
EVO	Engineering viewpoint ontology
FAT	Functionality acceptance test
FBS	Function-behaviour-structure
FEA	Finite element analysis
FEM	Finite element method
F-logic	Frame logic
FO	Foundation ontology
GFP	Generic frame protocol
GT	Glossary of terms
GUI	Graphical user interface

HTML	Hyper text markup language
ICI	Interface configuration instruction
IGES	Initial graphics exchange specification
IP	Intellectual property
IR	Information retrieving
ISO	International Organization for Standardization
KA	Knowledge acquisition
KB	Knowledge base
KBS	Knowledge-based system
KIF	Knowledge interchange format
KIM	Knowledge and information management
KM	Knowledge management
KR	Knowledge representation
LIMMA	Lightweight model with multi-layer annotation
LMV	Large model visualization
MCAD	Mechanical computer-aided design
MEV	Multiple specialist viewpoints
MPEG	Moving picture experts group
MSC	Measurable success criteria
NLP	Natural language processing
OCML	Operational conceptual modelling language
OGUI	OntoCAD graphical user interface
OKB	OntoCAD knowledge base
OKBC	Open knowledge base connectivity
OMA	OntoCAD MEV agent
OntoCAD	Ontology-driven semantic annotation framework for CAD systems
OWL	Web ontology language
PCE	Product cost estimation
PDF	Portable document format
PDM	Product data management
PLC	Entire product life cycle
PLM	Product lifecycle management
PMI	Product and manufacturing information
PPM	Product portfolio management
PSRL	Product semantic representation language
PSS	Product-service-system
Rbox	Relation box
RDF	Resource description framework
RDFS	RDF schema
RML	Rule markup language
SC	Success criteria
SFS	Semantic file system
SHOE	Simple HTML ontology extension
SI	The International System of Units
SIL	Semantic instead of location
SME	Small and medium-sized enterprises
SQL	Structured query language
SQWRL	Semantic query-enhanced web rule language
STEP	STandard for the Exchange of Product model data
SW	Semantic Web
SWRL	Semantic web rule language
TA	Transformation agent
Tbox	Terminology box
UDO	User defined object
URI	Web uniform resource identifier

VR	Virtual reality
VRML	Virtual reality modelling language
W3C	The World Wide Web Consortium
WBS	Work breakdown structure
XIRAF	XML information retrieval approach to digital forensics
XML	Extensible markup language
XOL	XML-based ontology exchange language

Related Pulications

1) Li, C., C. McMahon and L. Newnes (2009). Annotation in Design Processes: Classification of Approaches. International Conference on Engineering Design, ICED'09, Stanford. (Status: Published.)

2) Li, C., C. McMahon and L. Newnes (2009). Annotation in Product Lifecycle Management: A Review of Approaches. 29th Computers and Information in Engineering Conference (CIE), San Diego, ASME. (Status: Published.)

3) Li, C., C. McMahon, L. Newnes and Y. Liu (2010). Ontology-Based Annotation in PLM Systems. International Conference on Product Lifecycle Management, Bremen, Germany, Inderscience Enterprises Ltd. (Status: Published.)

4) Li, C., C. McMahon and L. Newnes (2011). Progress with OntoCAD: A Standardised Ontological Annotation Approach to CAD Systems. International Conference on Product Lifecycle Management, Eindhoven University, The Netherlands. Inderscience Enterprises Ltd. (Status: Published.)

5) Li, C., C. McMahon and L. Newnes (2011). "OntoCAD: A Semantic Annotation Approach to Support Multiple Engineering Viewpoints in CAD Systems." Advanced Engineering Informatics. (Status: Submitted.)

Chapter 1 Introduction

This research work focuses on knowledge and information management (KIM) in the engineering context through the entire product life cycle (PLC) (i.e. from initial product design until its disposal). Knowledge and information from multiple engineering viewpoints (MEV) often need to be developed and incorporated together through engineering tools in a collaborative environment, such as geometric modelling, manufacturing, engineering analysis, and so on. The multiplicity of tools and viewpoints means that achieving such collaboration is challenging, especially concerning the integration and interoperability of tools. In order to further assist with knowledge representation (KR), interoperability and tool integration in the PLC, new approaches are needed, especially at the late design and manufacture stages after computer-aided engineering (CAE) are involved, including computer-aided design (CAD) and computer-aided manufacturing (CAM). The research presented in this thesis focuses on one such approach: semantic annotation to aid the engineering design process within the domain of mechanical engineering.

Typically, engineers spend the majority of their total work time on data handling related activities (89%) rather than creative activities (11%), including administration, communication, decision making, approval, and information management (Rangan et al. 2005). During such activities, a great deal of development involves the re-use and re-processing of existing data, instead of creating new data (Buneman et al. 2005). This implies that information management needs to play an important role in enabling more effective re-use/re-processing of data during the entire PLC.

This is emphasised by the shift in business process models from the one-off sale of products to the provision of products and associated functions or services. Firms are thus taking more responsibility for products during their entire PLC in order to run sustainable businesses to cope with global competition (Mont 2001; Sharma 2007). To meet this industrial need, product lifecycle management (PLM) has been introduced as an systematic concept to support the development and management of product related definitions (Saaksvuori and Immonen 2008). Its corresponding computational enablers, PLM systems, are developed to serve the entire PLC as a mechanism for representing, storing and sharing, collaborating, visualizing product information and so forth (Stark 2011). PLM systems have developed rapidly in the last two decades, and are still attracting much attention. According to CIMdata (Farish 2008), the gross value of the world PLM market was predicted to reach $39bn by 2012, increasing from $24.3bn in 2007. In such a complex system, information management, as a key element, plays a

very important role throughout the PLC, including information storing, retrieving, communicating and interpreting. In this research work, three major challenges in the development of PLM systems are identified and are addressed, namely:

- Knowledge management including knowledge representation (e.g. associativity and data structure);
- Incorporation of MEV;
- Knowledge interoperability and system extendibility.

The first major challenge is knowledge management, in which the associativity between general purpose engineering information and design models needs to be addressed. In the mechanical engineering domain, CAD systems, as an important constituent of PLM systems, play a critical role during design activities (McMahon and Browne 1998). However, they are weak in capturing and managing product information to be associated with design model for general purposes, especially input from users. For example, according to the author's observation on commercial CAD systems (e.g. Siemens' NX and Dassault Systèmes' CATIA), the users have limited flexibility to attach general purpose information to design models and limited query capability to retrieve the information from the models.

With regard to knowledge management, another issue can be to address collaborations among MEV. There are efforts on neutral formats for geometric representations to support such collaboration, such as the ISO[1] 10303 STandard for the Exchange of Product model data (STEP) (Pratt 2001; El-Mehalawi and Allen Miller 2003a; Pratt 2005), or product and manufacturing information (PMI) as a collection of product definition and manufacturing information (Korneffel and Dvorak 2004). However, it is still an open issue to manage wider engineering information within a single CAD system (Vijay 2011). Furthermore, it is difficult to exchange data between CAD systems and their external environment (Kim et al. 2008; Kim et al. 2010; Song and Han 2010). This challenge creates obstacles to efficient design information reuse, data and tool integration, which is mainly caused by insufficient knowledge representation schemes.

The second major challenge is the semantic interoperability among collaborating knowledge domains. It has been reported that inadequate interoperability in the U.S. capital facilities industry costs at least $15.8 billion per year (Gallaher et al. 2004), and

[1] ISO stands for the International Organization for Standardization.

also costs more than $1 billion each year within the U.S. automotive supply chain, with the major portion existing in data reprocessing, such as data file repairing and re-entering (Brunnermeier and Martin 1999). In particular, current CAD systems lack the capability to allow users to add more semantically rich information to the design models, which is important to describe design models explicitly in order to avoid ambiguity when sharing knowledge across domains and to improve the ability of knowledge processing automation.

Last but not least, the third major challenge is about the system extendibility. Apart from the knowledge representation, knowledge management, interoperability and exchange with external environment, the KMS itself should be extendable to adapt to environmental changes. For example, a PLM system is a toolset that incorporates services for products to cover the entire PLC (Saaksvuori and Immonen 2008), but it comes with high cost to integrate new tools or legacy tools, mainly due to their incompatible and heterogeneous data infrastructures (Brunnermeier and Martin 1999; Ball et al. 2008; Silcher et al. 2010). Therefore, it calls for an extendable framework, in which existing tools can be updated, and new tools can be integrated as a total system with low cost, including less labour, less programming effort, lower expertise requirements, short lead time and so on.

To overcome these challenges, some technologies are considered, such as file systems (Hung Ba et al. 2007; Eck and Schaefer 2011), database management systems (DBMS) (Silberschatz et al. 2010), and process and meta modelling (Rolland 1998; Grüninger and Menzel 2003; Amelunxen et al. 2008; Weisemoller et al. 2008). However, file systems have limited high level granularity to information objects, database systems lack semantics, and process modelling technologies do not cope with knowledge representation and integration very well. It is the author's view that investigating the use of semantic annotation will contribute to and enhance CAD systems. Annotation is the extra information inserted in particular place(s) of an original document (Ovsiannikov et al. 1999). For example, annotation may be used to add notes onto the CAD model (e.g. design intention and manufacturing constraints). However, for users to add extra information the method needs to be manageable and accessible. Furthermore, to reuse the extra information, especially in more intelligent ways, new approaches are desired. With the support of ontologies, annotations can be endowed with semantic features, therefore are capable of representing knowledge. This opens the possibility of deriving new knowledge by reasoning over existing knowledge (Staab et al. 2001; Stephan et al. 2007).

This research aims to explore and develop a semantic annotation approach that assists

with information management and its downstream processing based on CAD systems. This will allow improved collaboration among different fields of expertise, information interoperability and tool integration. To achieve this aim, a systematic approach is proposed to represent and control this additional information within CAD models. This proposed approach will further assist with information and design reuse, as well as specifying a formal methodology for knowledge modelling. This approach will be based on a knowledge-based system (KBS) architecture, where an important computational enabler – ontology technology - is deployed as the heart of a knowledge base (KB). An ontology (See Section 4.2) is a concept used to represent knowledge in an interested area by explicitly specifying an abstract view of the world (Gruber 1995).

1.1 Aims and Objectives

The aim of the research presented in this thesis is to explore and define a systematic semantic annotation framework to assist with CAD-based design in the domain of mechanical engineering, where annotations are used to represent and manage engineering knowledge in supporting MEV, are able to derive a data model and enable downstream processing. To achieve this overall aim the following specific objectives were identified.

Objective 1 To explore the literature related to PLM, engineering design, CAD, MEV and then the related technologies and the applications in various domains, with an emphasis on mechanical engineering. Thus to identify research gaps in engineering knowledge management.

Objective 2 To define an extendable framework to systematically manage knowledge in supporting MEV in order to assist with knowledge acquisition (KA), knowledge representation, data and tool integration and interoperability.

Objective 3 To provide a data structure that is able to accommodate knowledge and allow it to be associated with design objects.

Objective 4 To define a mechanism that supports the query and exchange of data, and to derive new knowledge based on existing knowledge.

Objective 5 To provide guidelines for a rigorous methodology to assist with knowledge modelling to allow the proposed framework to be maintainable and extendable.

Objective 6 To design and develop a demonstration system, and thus to evaluate the success or otherwise of the proposed framework in terms of the feasibility,

effectiveness and generality.

These objectives will be expanded in the following chapters. Although there are many other research challenges in current CAD systems and information technologies (e.g. security, natural language processing, etc.), they are out of the scope of this research. To address the overall research aim and key objectives, a sound research methodology is essential. The research methodology utilised in this research is described in Chapter 2 along with the thesis structure, which reflects the methodological approach adopted.

Chapter 2 Research Methodology

In general, a methodology is a system of methods that is applied within a particular area of study or activity (Soanes and Stevenson 2005b). In the context of design research, the aim of a methodology is to assist in formulating, developing and validating the research in order to refine design practice, management, education and their outcomes. In a typical design research study, the outcomes include the understanding of a model or theory of the existing situation, a derived model or theory of improvement, and the necessary support that leads to the improvement. A rigorous methodology increases the chance that research will lead to better transformation from research results into practice, better evaluation of research, and more effective advancement of the research (Blessing and Chakrabarti 2009).

In order to guide the selection and application of a suitable approach and appropriate methods, and to encourage reflection on the approach and methods to be used during the research lifetime, research methodologies were explored and a suitable one adapted for the need of the research presented in this thesis. The adapted methodology is then used to introduce the thesis structure and to illustrate how each chapter represents the stages of the methodology selected.

2.1 Generic Research Methodologies

In the context of social science, Creswell (2009) defines that research methodologies can be classified into *qualitative*, *quantitative* and *mixed methods* approaches. These approaches are differentiated by their knowledge claims, strategies of inquiry and methods of data collection and analysis for their emphasized purposes. Qualitative approaches reflect the research associated with strategies such as grounded theory, case studies, phenomenological research, and are often used in the social sciences domain where subjective data is utilised. Quantitative approaches are associated with experimental work (e.g. experiments, surveys etc), which are more suitable for identifying factors and evaluating theories. To some extent the mixed methods approaches are combinations of qualitative approaches and quantitative approaches. Qualitative and quantitative approaches compliment each other. In the engineering field, quantitative approaches are often more appropriate and were therefore utilised in this research.

Some general design research methodologies include Bracewell et al. (2001), Duffy and O'Donnell (1999), Brinkkemper (1996), and Hevner et al. (2004). In these, similar processes are proposed: design problem; hypothesis; research problem; solution; formal

evaluation; and documentation. However, their proposed frameworks or guidelines provide limited information to lead research work to identify and solve problems.

To provide such guidance, the design research methodology (DRM) (Blessing and Chakrabarti 2009) presents a more detailed approach that has four stages: research clarification (RC), descriptive study I (DS-I), prescriptive study (PS) and descriptive study II (DS-II). RC clarifies the understanding of current theory or models and derives an overall research aim with a plan. The following stage, DS-I, gains a further understanding of the design issue and identifies the effective factors necessary for improvement. Once a sound understanding has been gained, a solution with necessary support is defined in the PS stage, based on the previous study. The final stage, DS-II, evaluates the solution and its support in terms of usability, applicability and usefulness etc.

In practice, the research processes at each stage can be different types of studies, suggested by Blessing and Chakrabarti (2009): the review-based study, comprehensive study and initial study. The review-based study is based on literature review only. The comprehensive study requires review and analysis of both literature and results produced by researchers themselves, including empirical study, evaluation of results. An initial study concludes a project and prepares the results for future use. In general, the numbers of stages and the types of studies may vary according to the needs of different research projects. For this research the DRM framework has been adapted to follow the four stages with various types of studies

2.2 Adapted Research Methodology

Based on the aim and objectives stated in Section 1.1, the DRM was adapted to suit the needs of the presented research as shown in Figure 1. There were six stages in this adapted framework with medium complexity and iterations.

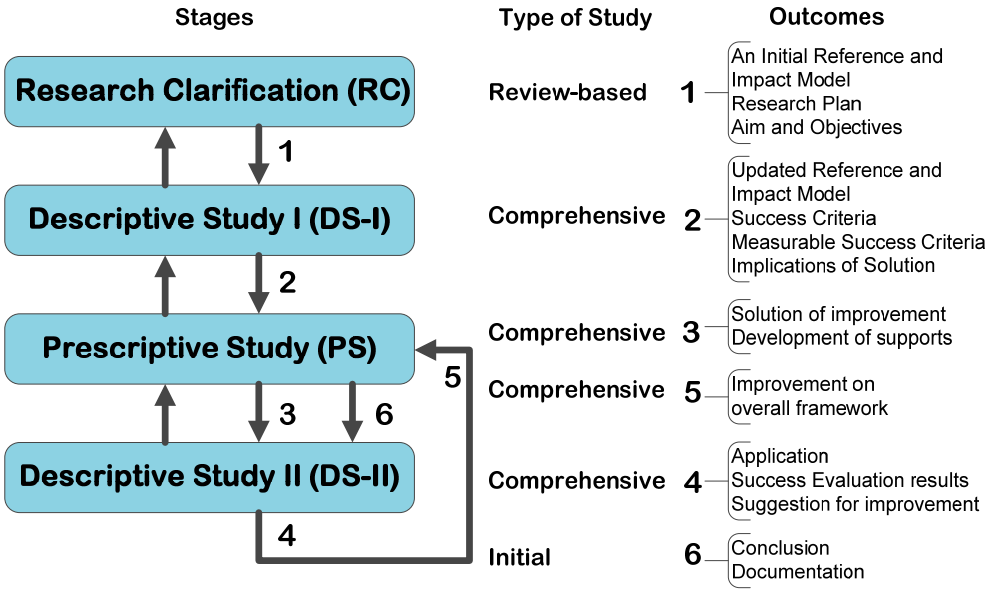


Figure 1 Adapted DRM Framework (Blessing and Chakrabarti 2009)

Research Clarification (RC)

In the first stage RC, the primary objective is to identify the goals that the research is expected to realise. This includes the scope of the research project, the main research questions and hypotheses; the related literature to be reviewed, and the potential contribution. Based on this review-based study, the deliverables were:

- Initial understanding and prospect.
- Initial reference model (RM) and impact model (IM).
- Preliminary success criteria (SC) and measurable success criteria (MSC).
- Overall research plan with time schedule.

The reference model (RM) refers to the existing situation of CAD-based annotations, while the impact model represents the desired situation. An example statement is illustrated in Figure 2. At the RC stage, the initial situations were expected, updated and refined along the development of the research work. In some cases, RM and IM were combined for the purpose of conciseness.

The success criteria are success factors that are used to evaluate the ultimate goal of the research. The measureable success criteria are reliable factors that can be linked directly or through reasoning to success criteria in order to judge practically the outcome of the research. Among the factors, key factors are the root causes that can most effectively improve the existing situation.

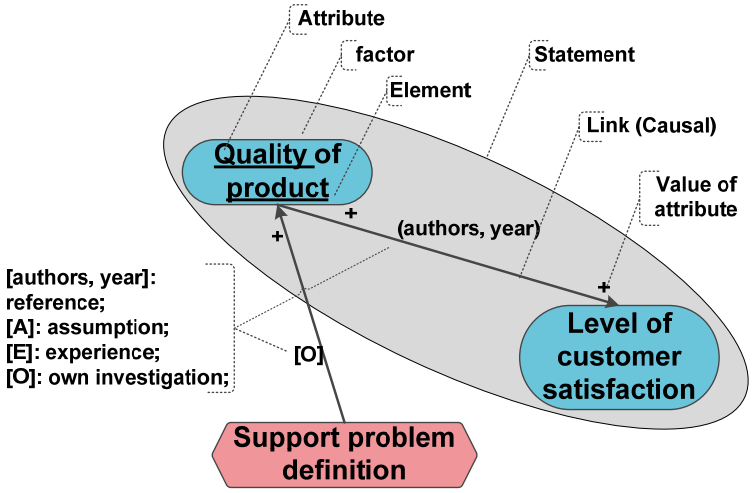


Figure 2 DRM Representation of a Statement and Modelling Terminology

Descriptive Study I (DS-I)

The DS-I stage aims to gain a more thorough understanding in the phenomenon of design, namely the state of the art of CAD systems and its computational enablers through comprehensive study, including literature review and empirical research. This refers to identifying and clarifying in more detail the effective factors that highlight the existing situation and problems, thus to improve reference and impact models together with the improved criteria. In order to establish a basis for developing a solution at next stage, the deliverables at this stage were:

- A completed reference model, an updated impact model, SC and MSC.
- Implications of the findings to conduct a solution and the development of support for evaluations.

Prescriptive Study (PS)

The ultimate goal of a research project is to develop a support that improves the existing situation, which happens at the PS stage. The objectives are to define a solution (*intended support*) for such improvement in an ideal situation based on the understanding gained from the RC, DS-I or/and possibly from DS-II stages, and also to develop an *actual support* (e.g. prototype software system) in order to realize and evaluate the core concept with limited functionality, robustness and coverage. Based on the comprehensive study, the deliverables were:

- Documentation of the Intended Support, including an intended impact model.

- The impact model, the actual support and its documentation.
- The verification of the actual support (support evaluation) to ensure the prototype is correctly developed.

Descriptive Study II (DS-II)

The DS-II stage focuses on evaluations based on the support developed in the PS stage. There were mainly two types of evaluation: application evaluation and success evaluation. The application evaluation focuses on usability and applicability, to investigate if users are able to understand and use the support, and whether the quality needs to be improved. The success evaluation more focuses on usefulness against the measureable success criteria, e.g. how effective the proposed solution is. Since it was not aimed to develop a commercial off-the-shelf (COTS) application and with limited resource on evaluating user experience, it was more focused on the success evaluation in this thesis. Based on this comprehensive study, the deliverables were:

- Outline of evaluation plan based on previous study.
- Results of the success evaluation and application evaluation.
- Implications and suggestions for further improvement.

The Iterations – Refinement of the Solution

In this research, there were minor iterations for an optimized solution. Apart from the technical aspect, the maintenance and evolvement of the proposed system were also considered. This implies the need for a systematic knowledge modelling methodology. And it was necessary to iterate previous stages to evaluate whether the modelling methodology was appropriate. Since there are many knowledge modelling methodologies available, without experiments it can not be confirmed whether they can fit in the proposed framework. These iterations aimed to evaluate and refine the knowledge modelling methodology in different situations: modelling knowledge from scratch, updating a knowledge module, and so on. The actual development of the knowledge modelling methodology has gone through iterations between PS and DS-II (Stage 5 and 6 as shown in Figure 1). However, it will be reported in one go in order to be concise.

2.3 Structure of the Thesis

This thesis is presented in nine chapters. Each chapter is briefly described and mapped to

its corresponding DRM stages as depicted in Table 1.

Table 1 Structure of the Thesis with Mapping to DRM Stages

Chapter Number	DRM Stage	Basic Methods	Brief Descriptions
Chap. 1	RC	Literature review	The chapter includes a brief introduction of this thesis, and also the motivation, aims and objectives and scope of the research.
Chap. 2	RC	Literature review	It introduces research methodology adopted in the work and the structure of this thesis.
Chap. 3	RC	Literature review and empirical study	This is the first part of background research mainly based on literature review and empirical confirmation. It briefly gives an introduction to foundation knowledge of PLM, CAD, the concept of MEV, and the case studies used in evaluations. This chapter also describes the state of the art in these areas. Based on this study, semantic annotation is concluded as a promising computational enabler.
Chap. 4	DS-I	Literature review and empirical study	This chapter focuses on the introduction to the identified key computational enablers including annotation and ontology that potentially address challenges described previously. This chapter also describes the state of the art of these technologies and the applications in different fields, thus concludes that some findings that implies a possible improvement on current situation.
Chap. 5	PS	Literature review and empirical study	The framework OntoCAD is proposed in this chapter, where research objectives are revisited, scope and focuses of this research are explicitly defined. As key modules, annotation data structure, the knowledge base kernel and knowledge modelling methodology are described in detail.
Chap. 6	PS	Literature review and empirical study	As actual support to the OntoCAD concept proposed in the previous chapter, a prototype software application is designed, developed and documented in this chapter, in which an evaluation plan is also specified.
Chap. 7	DS-II	Empirical study	Two case studies are carried out and described in this chapter to evaluate the usefulness of the proposed framework. The knowledge modelling methodology based on the observation of two case studies is also testified.
Chap. 8	DS-II	Initial study	The results from all previous studies, viz. the RC, DS-I, PS and DS-II are reviewed, analysed and discussed. The limitations and advantages of the proposed OntoCAD framework are concluded. Based on the results, some possible future work is suggested.
Chap. 9	DS-II	Initial study	This chapter summarises the research presented in this thesis.

Chapter 3 Background – Part I

This chapter describes the overall context of this PhD research. As introduced in Chapter 1, the areas of relevance and contribution (ARC) are mainly focused on knowledge and information management in the mechanical engineering context with particular interest in the topics of engineering design, computer support and engineering collaborations, as shown in Figure 3.

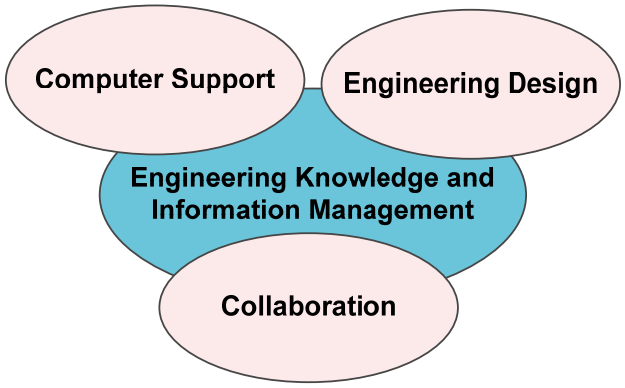


Figure 3 Areas of Relevance and Contribution Diagram

Although it is believed that the work presented in this thesis can be extended to assist with engineering services within PLM (and this will be discussed in Chapter 8), CAD as one computational support to PLM is the primary focus of this research. Multiple engineering viewpoints (MEV) are also introduced as a key concept that helps knowledge management (KM) through the means of collaborating with knowledge from various domains. In this section the state of the art of each topic is also explored. In addition, two case studies – cost estimation (CE) and finite element analysis (FEA) are also described. The two studies are used to identify the research challenges in the current engineering environment, and will be used for evaluating the outcome of this research. This chapter is concluded with some key findings on current research gaps of PLM and CAD that lead to further study in the next chapter.

3.1 Product Lifecycle Management (PLM)

PLM is a series of business activities of managing a product or service during the entire PLC including portfolio management, product design, process design, supply, production, product launch, service and support, end of life and recycling (Stark 2011). To aid such series of activities, PLM systems have emerged as a systematic toolset that provides wide functionality to support the management activities for product definitions during the entire PLC (Kiritsis et al. 2003; Sääksvuori and Immonen 2008).

PLM and its implementation has been developed over decades and has effectively

contributed to reducing cost, improving quality, shortening time-to-market and assisting with innovation (CIMdata 2002). In the late 1990s, product data management (PDM) gained popularity, in other words it is to manage the data related to a particular product with the focus transferred from individual companies to companies' supply chains. PDM mainly aided in the management and publishing of product data, including the information related to design geometry, engineering drawings, project plans, product specifications, analysis, bills of material, and many other items. Therefore, PDM systems have some overlap with CAD systems in regard to historically playing a role of storing design documents including CAD models, and later on being able to interact with CAD models through bidirectional data exchange (Rangan et al. 2005).

Based on PDM, its successor PLM thereafter was developed to cover the services throughout the PLC by fulfilling the five core functionalities (Burkett et al. 2002):

- PDM that mainly aids the publishing and management of product data;
- collaborative product design (CPD) that assists with product design and manufacturing process design;
- direct material sourcing (DMS) that supports the data handling between suppliers and vendors;
- customer needs management (CNM) that mainly serves customers;
- product portfolio management (PPM) as a general service to all participants.

PLM systems can also be classified into two types: one has a focus on product design and one emphasises collaborative data management (Cheung and Schaefer 2010). In both types, efficient communication and seamless collaboration play a vital role in successful enterprises (CIMdata 2002), since business often needs to be carried out between different teams within a single company or among cooperative companies in a geographically distributed environment throughout PLC. Here, each participant may hold different viewpoint to an identical product.

3.1.1 The State Of The Art In PLM Systems

Currently, the development of PLM systems has reached an initial state of maturity. There are many commercial PLM systems (PLM Technology Guide 2008), including some leading ones that cover most areas of PLM functionalities: Agile (Oracle 2011), Enovia (Dassault Systèmes 2011b), ProductCenter® (SofTech Inc 2011), Teamcenter (Siemens PLM Software Inc 2011b), and Windchill (Parametric Technology Corporation 2011b).

As stated previously, the fast growing market of PLM is still gaining more attention. Many research issues have been raised. According to Cheung and Schaefer (2010), forty five PLM systems are analysed in terms of support of databases, operating systems, industry standards, target company size, and customizing scripting languages. The authors conclude that customization to fit users' needs (e.g. enterprise size or available scripting skills) and IT infrastructure are the main requirements in adopting PLM systems.

Ming et al. (2005) suggest some research gaps in current PLM capability, including CAD/CAM integration versus collaboration for product development and real time design; product planning versus product portfolio management; part repository for reuse versus product lifecycle knowledge management; product and part maintenance versus extended product service. Rangan et al. (2005) and Hewett (2009) also point out that evolvement directions for PLM systems have been data exchange, design collaboration, enterprise-centric view, scale to reality, standards and techniques for engineering processes, information and knowledge representation. A future vision by Terzi et al. (2010) is that PLM systems can provide a closed loop of information, in which a complete data model of products can be up to date in real time, expertise can be exploited, information can be explicitly accessed, the system can be sustainably maintained.

Based on the findings from literature, it can be concluded that resource requirements for deploying PLM systems, data integration, design collaboration and knowledge processing are the main challenges. Research efforts have been repeatedly dedicated, however issues still remain (Hewett 2009; Terzi et al. 2010).

Resource Requirements

One of the significant issues is the resource requirements for adopting PLM systems. PLM systems have been adopted in many key industries, including aerospace, defence and automotive industries and so on, especially by large scale enterprises that are valued higher than $1billion (Cheung and Schaefer 2010). Some companies have also developed in-house systems to meet their own specific needs, e.g. VIPER launched by Heinz (Brown 2003). However, PLM systems are normally not implemented by small and medium-sized enterprises (SME). This is because the resources required by the PLM systems are very expensive, due to the nature of the complexity of the systems. Ideally, a generic flexible PLM system is needed, which can be tailored to most customers' needs without high adoption cost, including the price of PLM systems and resources required for implementation.

Data Integration and Collaboration

Another issue is the data integration and collaboration among participants. Since there are many commercial PLM systems and standards available and proliferating due to the diversity of data and its distribution, users need to choose their own system and standards (Terzi et al. 2010). For a successful PLM system, it needs the ability to handle the multiplicity of document formats in order to integrate newly developed applications or legacy tools, however this still remains challenged (Ball et al. 2008). For example, this situation may cause data loss during data exchange between PLM systems. If an user of a CAD tool NX (Siemens PLM Software Inc 2011a) exports a design model into the virtual reality modelling language (VRML)[2], the design history information (i.e. the history of how the design model was created) may be lost, such as geometric modelling history and user annotations.

Although all existing PLM systems try to address design collaboration, it is still hindered by limitations in the various industrial standards and data formats. Data exchange between CAD tools often requires an intermediate format such as the STEP (LAMP/IDE 2008) or the extensible markup language (XML) (Bray et al. 2008). Many research efforts try to address the transformation between standards (Peak et al. 2004; Beetz et al. 2005; Krima et al. 2009). Unfortunately, not all design detail can survive during the format transformation, such as modelling history, modelling features or granulation. The weakness in product data infrastructure further affects other issues, such as downstream data processing (e.g. data retrieval and traceability), distributed collaboration and so on (Ball et al. 2008),

3.1.2 Concluding remarks

This section has given an introduction to PLM, its brief history and the state-of-the-art in commercial application development. Some research challenges and directions are also identified, such as deployment requirements, data integration and design collaboration. These challenges are also shared by CAD systems. To some extent, CAD systems either interact with PLM systems (Rangan et al. 2005) or have been integrated as an important part of a PLM environment (CIMdata 2002). In the next section, engineering design and CAD are introduced and also their current status is reviewed.

---

[2] VRML is designed to describe 3D objects for Web resources with an interactive ability. (Bell et al. 1995)

3.2 Computer-Aided Design

In this section, the history and development of engineering design processes are reviewed, followed by the introduction to the history, state-of-the-art of CAD systems, and some key technologies, including three-dimensional (3D) geometric modelling technologies. The gaps in current CAD systems are then identified. This analysis provides the foundation for the research presented in this thesis.

3.2.1 Introduction to Engineering Design

A widely recognized engineering design process comprises four main stages, i.e. clarification of the task, conceptual design, embodiment design and detail design (Pahl et al. 1984). In the clarification stage the design specification is defined through collecting requirements for and the constraints on the design. The conceptual design stage establishes functions and the designers propose and choose design solutions. Within embodiment design a higher level of maturity on solutions by solving problems and weaknesses is achieved. Finally, the detail design stage involves defining and refining all components of a project in full detail, in terms of the dimensions, tolerances, and materials and so on. Within each of these stages and between them, reviews and evaluations are normally carried out iteratively until a satisfactory solution is reached. In engineering design, communication plays an extremely important role for participants to collect information, share comments and authorise and approve designs so that to collaborate among product designers, manufacturers, analysers and end users in any design stage (Rangan et al. 2005; Pahl et al. 2007).

In order to be more competitive and keep pace with the ever changing market, there is pressure to shorten the time-to-market, reduce cost etc. For this reason, concurrent engineering has been introduced to strengthen productivity at lower cost and shorter product lead-time (Brookes and Backhouse 1998; Kusiak and Larson 1999). Again, communication and collaboration between participants in any design process are even more critical (Sonnenwald 1996).

In practice, systems for CAD, CAM, computer-aided process planning (CAPP), and other CAE are widely used in aiding engineering design, especially in the process of detail design.

3.2.2 Introduction to CAD

As a result of the advances in computer science over many years, CAD has been

developed as an approach for modelling and communicating to aid productivity and automation in engineering design process (Salzman 1989; Robertson and Radcliffe 2009). A typical CAD system normally consists of hardware, software, data, human knowledge and activities (McMahon and Browne 1998). The hardware includes the computer and associated peripheral equipments, e.g. some specialized printer or scanners; the software is the computer programs, which is gradually becoming more and more complex; the data are created and managed through the software, normally associated with a database management system. Other than engineering design, CAD systems are also widely used in other fields, such as medical care (Strub et al. 2006), construction (Shen et al. 2007), and garment design (Liu et al. 2010) and so on (Tornincasa and Monaco 2010). In this research work, mechanical engineering design will be the core focus.

During the engineering design processes, CAD systems often play a critical role in terms of providing a formal process of creating drawings, designs and models in many engineering design (McMahon and Browne 1998). Within the design domain there are two predominant modelled properties, namely, form (shape) and structure. To define the form and structure, many 3D modelling schemes have been developed over the last forty years, such as; wire-frame geometry, the surface representation scheme, and solid modelling (Woodwark 1986; Anand 1993; McMahon and Browne 1998).

The geometry in the wire-frame scheme appears in a wire-like manner, defined by a series of lines and curves, while in the surface modelling scheme, geometries are defined partially or entirely by flat or curved surfaces on a component. Solid modelling is currently one of the major modelling schemes and utilises two main methods. One is constructive solid geometry method (CSG or C-rep), which uses set theory such as union, intersection and difference operators to combine simple solid primitives, e.g. cuboids, cylinders and cones. The other one is the boundary representation method (B-rep), which dominates in current CAD systems, where a solid body is represented by its bounding faces, edges and vertices and so on (Ault 1999; Babic et al. 2008).

Regardless of the modelling schemes, the underlining modelling techniques are all mathematics based. Geometries can be specified by defining parameters of curves, which is described using the concepts of Hermite and Bezier curves, Bezier-Bernstein polynomial functions, splines, and so on in mathematical terms. All varieties of surfaces are fundamentally based on these concepts (Woodwark 1986; Anand 1993; McMahon and Browne 1998).

Since CAD systems are widely deployed as common practice in engineering design (McMahon and Browne 1998), they have been well developed as commercial applications, and also have attracted intensive attention in academic research. The next sections will describe the state-of-the-art in both aspects.

3.2.3 The State of the Art in Commercial CAD Systems

In industry, CAD systems are normally delivered to the customer as CAD system software packages, in which many design and drawing tools are provided. CAD systems have evolved from two-dimensional (2D) drawing systems to 3D modelling systems. The first CAD system "Sketchpad" was innovated by Ivan Sutherland (1964) at MIT in the early 1960s. In the late 1970s, CAD/CAM systems shifted to 3D drawing (Tornincasa and Monaco 2010), but 2D sketching is still supported in modern CAD systems. 3D systems are currently the most popular systems that offer most features, which allow users to do solid modelling or surface modelling. Some of the systems are capable of 3D parametric feature-based modelling, and tend towards extensive integration of applications including manufacturing and analysis (Tornincasa and Monaco 2010).

There are many CAD software package providers and products in engineering fields (Wikipedia® 2011b). Among them, the market of CAD systems is dominated by four vendors Dassault Systèmes, Autodesk, PTC and Siemens as listed in Table 2.

Table 2 Leading Commercial CAD Systems

Products	Vendors	Data Exchange Format Compatibility
Autodesk Inventor 2012	Autodesk Inc (2011b)	Alias, CATIA (V4 and V5), IGES, JT, NX, Parasolid, Pro/ENGINEER, Rhino, SAT, SolidWorks, STEP, STL, XGL/ZGL, and more.
CATIA V6	Dassault Systèmes (2011a)	CATIA (V4, V5, and V6), STEP, DXF, STL, VRML, IGES, and more.
Creo Elements/Pro (formerly Pro/ENGINEER)	Parametric Technology Corporation (2011a)	STEP, IGES, DXF, DWG, Parasolid, JT, ASIC, CADDS, CATIA (V4 and V5), NX and more.
NX 7	Siemens PLM Software Inc (2011a)	JT, Parasolid, STEP, DWG, DXF, Pro/ENGINEER, SolidWorks, I-deas, CATIA (V4 and V5), STL, IGES.

Standardization has been pursued for over two thousand years, for example currency, weights and measures were unified in the 2nd century BC in ancient China (Qing 1995). Standards are always critical to engineering design and this is particularly true in data

exchange between CAD systems (Björk and Laakso 2010). As Table 2 illustrates, there are currently many standards in data exchange across CAD systems. A survey by Tan (2006) reveals the proportion of data exchange standard usage in mechanical design and manufacturing industry in the area of Europe and North America (Figure 4). Among them, DXF (and its predecessor DWG), IGES, PDF and STEP are dominating standards for data exchange between different CAD systems.

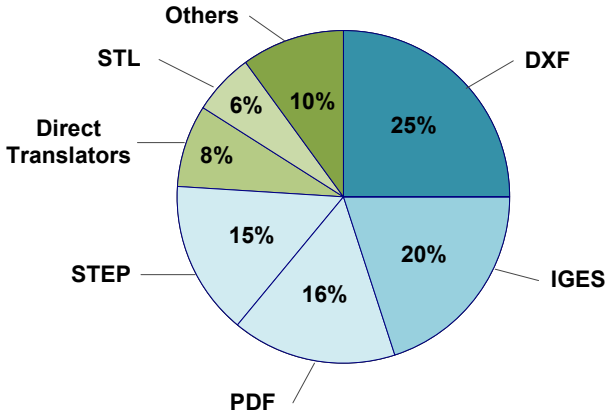


Figure 4 Data Exchange Standards for CAD Systems in Europe and North America (Tan 2006)

- DXF stands for drawing exchange format (Autodesk Inc 2011a), is a data exchange format for CAD systems that is proprietarily owned by Autodesk©, while DWG (drawing) is a CAD file format also owned by Autodesk© but without specification published.
- IGES stands for initial graphics exchange specification (NIST 2011). It is a neutral data format for 2D drawing and 3D model, which can be used to exchange design data between CAD systems. However, IGES tends to be replaced by the STEP standards due to its deficiency (Srinivasan 2008).
- The ISO 10303 standard (Pratt 2001), informally known as STEP is a standard family for exchanging product-related data between CAD systems and downstream application systems, which currently tends to cover entire PLC stages (Pratt 2005). STEP contains a set of parts, including application protocols (APs) that covers wide range of product types, such as ship, automotive, constructions and so on (ISO 1994a). The STEP standards are defined using a product modelling language called EXPRESS (ISO 1994b) (i.e. the schema). The Part 21 (ISO 1994c) defines the exchange format for encoding the product data.
- Portable document format (PDF) (ISO 2008c) is for document exchange specified by Adobe Systems Incorporated. One of its subsets, the PDF/E (ISO 2008b) (representing and exchanging engineering documents), provides the capability to handle engineering CAD drawing.

These standards can be categorized into two groups: proprietary and open standards. The evolvement of these standards indicates the open standardization is one of the important movements. This is interactively affected with the development of open-source CAD systems. There are many open-source CAD software (CADAZZ 2004; Wikipedia® 2011a), such as Archimedes, AutoQ3D, and Calculix. Both commercial and open-source CAD systems attracted immense interests, which will be described in the next section.

3.2.4 The Latest Development of CAD in Research

Due to the increasing popularity of CAD, much engineering design work is carried out using CAD systems (Robertson and Radcliffe 2009). As a result of growing complexity of engineering problems, using geometric modelling only no longer adequately suffices designers. CAD systems need to offer more intelligent functionalities to assist with decision making during the design process, by associating more information with the 3D forms, such as dimension, tolerance, material, and component functions from all engineering viewpoints. Some extra functions provided by modern CAD systems try to improve to the capability for knowledge representation and process automation, collaboration and interoperability.

Automatic Knowledge Processing

Automation here refers to automatic knowledge processing, in which artificial intelligence (AI) has been playing an important part. The knowledge implies the descriptions of the world that enables an intelligent machine to compute new conclusions within its context (Stephan et al. 2007). These descriptions can be stored in a knowledge base for sharing and reusing by KBSs, thus to aid in design automation (Colombo et al. 2007), for example, automatically complying with design constraints to modify its geometric dimensions. KBSs may also assist with design analysis or optimization by processing the captured multidisciplinary expertise, for example, to assist with automatic manufacturing process planning or selection of materials.

Another intelligent concept used in modern CAD systems is features (Shah and Mäntylä 1995). A feature is any perceived generic geometric or functional element or characteristic of a product useful in understanding the function, behaviour or performance of that object (Brown et al. 1992; Shah and Mäntylä 1995), such as drilled holes and grooves in shafts. Features can be used to assist with geometric modelling, known as feature-based modelling, which provides functionality for improving design automation and efficiency in

modern CAD systems. To automatically identify features in a design model using computer programs is known as feature recognition, which is desired in aid of analysis and decision making. However, while feature recognition is easy for a trained human observer, it is considered problematic to computers, in terms of inability to learn, limited range of recognition, low speed, etc (Ding and Yue 2004). There is much current research contributing to feature recognition. Brousseau et al. (2008) present a method to automatically generate feature recognition rules by applying learning algorithms based on training examples. Other efforts include an approach for freeform surface CAD models by Sunil and Pande (2008), an approach based on STEP format by Rameshbabu and Shunmugam (2009), and many other automatic feature recognition approaches reported by Babic et al. (2008).

Knowledge Representation (KR)

One of the keys to AI technologies is knowledge management (KM). Knowledge management involves knowledge storing (i.e. to encode knowledge into a suitable computing format), knowledge retrieval (i.e. to find knowledge as needed), and reasoning (i.e. to compute conclusions, inferences and explanations by an intelligent program), while the most important prerequisite is the knowledge acquisition and knowledge representation (Gašević et al. 2006a). To some extent, the level of automation of knowledge processes is decided by the degrees of formality, process-ability and expressiveness of such representations (Webster 1988), which currently is still an active research topic.

A potential solution to these challenges is semantics. Semantics can be broadly defined as the meaning associated with a terminology in a particular context (Patil et al. 2005), i.e. the adoption of techniques to formally express information objects more meaningfully in order to improve the process-ability. Hoffmann (2005) suggests that semantics is one of the future directions of CAD system development. Semantics make semantic modelling possible, where the CAD design and the design processes more meaningful in terms of parametric representations, shape design based on constraints and feature modelling.

Furthermore, an exploration in improving virtual reality (VR)[3] capability through the use of semantics was carried out by Toro et al (2006). The authors proposed a semantic

[3] Virtual reality implies the computer generated environments that can simulate physical presence in the real or imaginary worlds. (Vince 2004)

enhancement approach for VR in CAD systems. In order to achieve faster VR and less computational resources, geometric models are re-rendered by identifying and replacing similar geometric elements with semantically described substitutions from a predefined catalogue.

Interoperability and Collaboration

Hoffmann (2005) suggests that interoperability across different CAD systems is another direction desired by customers in order to confront difficulties due to the CAD suppliers using proprietary CAD data formats which causes problems in exchange of data and translating existing designs into another CAD system. Also the absence of semantics that record design histories, making it more difficult to transfer between CAD systems (Hoffmann 2005).

With the increasing complexity of CAD systems, lightweight CAD systems are preferred in some situations. For example, to aid collaboration between geographically distributed teams where network bandwidth is insufficient for data transfer, a document-driven design (DDD) approach was proposed for distributed CAD services in a service-oriented architecture (Wang and Nnaji 2006). This DDD mechanism automates the process of geometry generation in a batch mode by processing text documents. The documents describe the feature-based modelling process through lightweight clients. Its semantic features strengthen its ability of communication, data interoperability. It eases design reuse, reduces human errors, and also improves geometric model compression.

Ding et al. (2009) proposed lightweight model with multi-layer annotation (LIMMA), which is a lightweight representation approach for CAD model that uses annotation and mark-up practices to build the association between product data and geometric product definitions during the PLC. In comparison, the DDD approach addresses the automation of CAD modelling processes using a set of semantic instructions, while LIMMA attempts to record semantic information applied to a CAD model. The recorded information includes security information, tolerances, machining processes, surface finishes and so forth, from various specialist viewpoints.

Another weakness of current CAD systems is that they were traditionally developed as a standalone toolset, which has shortcomings for communication and collaboration between users in a real-time manner. Tay and Roy (2003) tried an approach called CyberCAD, which makes the use of broadband networks, enables real-time audio and video communication through geographically distributed CAD systems. CyberCAD also

enables users to collaborate by fetching and viewing a design part from other participants, thus to observe the progress made by the others. Unfortunately, CyberCAD is not yet able to handle simultaneous access to same target CAD model by multiple users.

Similar to CyberCAD, MUG (MultiUser Groups for conceptual understanding and prototyping) by Cera et al. (2002) is another collaborative CAD approach. MUG allows multi-users to start voice-over-IP conversation while collaboratively modelling a 3D geometry. One of its advantages over CyberCAD is that multiple users are allowed to modify a partitioned model at the same time, and propagate changes across all participants in real time. Another contribution of MUG is that it adds sharable semantics (e.g. descriptions of function and assembly flow) by annotating a CAD model. The semantic information is managed as structure-behaviour-function knowledge through the use of ontologies. But the annotations are made on CAD model as a whole, e.g. a part or a component, without access to further details such as surfaces and edges.

There is also much other research work on collaborative CAD, such as visualization-based design systems and the concise 3D representation schemes, 3D streaming technology etc. that are reviewed by Fuh and Li (2005).

Furthermore, there is another major issue on data exchange related to collaboration and interoperability. Traditionally, it requires the development of translation programs to exchange data between applications, in which case $O(n^2)$ translation programs are required if a translation program is developed between each pair of applications out of a total set of n (Schlenoff et al. 2000). In contrast, the Interlingua approach, in which translation is to and from an intermediate format, tends to be more widely adopted as only O(n) translation programs are required (Uschold and Gruninger 1996; Schlenoff et al. 2000).

In conclusion, many computational enablers including annotations, semantics and ontologies have been further developed and adopted in practice in order to address the previously stated challenges. These challenges have some parallel to the three categories of challenges suggested by Kasik et al. (2005): computational geometries, interactive techniques and scale (quality and quantity of data and its distributed network). In more detail, the challenge in computational geometries includes geometry shape control and design exploration through multidisciplinary evaluation, which are closely related to design automation; the interactive techniques refers to the capability of communication and collaboration among users and across different CAD systems that affect data interoperability and collaboration; scale may imply challenges in knowledge

representation reflecting to the integrated comprehensive or lightweight knowledge models to aid different purposes, such as to add semantic-rich information onto CAD models, or to enable downstream knowledge processing, or to improve the virtual reality.

3.2.5 Concluding remarks

In this section, computer-aided design has been briefly described from the concept of engineering design to CAD, the history and evolvement of commercial CAD systems, also the standards of data exchange across different CAD systems. At the end, the current research status of CAD systems are also described, in which some research directions and research challenges are identified, including knowledge management, the automation of knowledge processing, collaboration and interoperability. Among these challenges, knowledge management is indeed a prerequisite to all the others, which will be further described in the next section.

3.3 Knowledge Management and Multiple Engineering Viewpoints

In this section, KM will be further described in terms of knowledge acquisition, representation, storing, retrieving and possible downstream processing. And then the concept of MEV is introduced in order to aid knowledge management and therefore to aid engineering processes.

3.3.1 Knowledge Management

It is widely accepted that knowledge is the fundamental driver to the success of a company and to support all its business activities (Wiig 1997). It is understanding of a subject area (Durkin 1994), that includes the concepts, facts and relationship of the subject area, and how to manage to solve problems (Gašević et al. 2006a). Knowledge is authenticated information that differentiates from data (raw numbers and facts) and information (processed data) (Alavi and Leidner 2001).

KM is to use systematically and technological mechanisms to manipulate intellectual assets of an organization to solve problems (McMahon et al. 2004). To some extent, KM is to manage effective knowledge processes (EKP) (Wiig 1997). As previously introduced, typical KM processes include knowledge acquisition and representation, storing, retrieval and use. In each process, mechanisms and technologies are needed to support them.

There are many different knowledge perspectives. Alavi and Leidner (2001) suggest that these include meta-knowledge (personalized information), state of mind (knowing and

understanding), object, process (to apply expertise), access to information (condition), and capability (the potential to influence action). In a more practically helpful classification, Michael Polanyi (1966) categorizes knowledge into two dimensions: tacit and explicit.

Tacit knowledge is rooted in action, commitment and involvement in a specific context. It has personal quality that consists of technical and cognitive elements. A technical element refers to informal personal skills of know-how, crafts and skills. The cognitive element refers to mental models encompassing schemata, paradigms, beliefs and viewpoints that help human beings to perceive and define their mental world. Explicit knowledge or codified knowledge refers to the knowledge that can be transmitted (i.e. articulated, codified and communicated) in systematic formal language (i.e. symbolic form and natural language) (Alavi and Leidner 2001). In a digital context, it can be in any recordable form, such as data, scientific formulate, specifications and manuals.

The methods required in KM processes are different depending on the knowledge perspectives. As Polanyi (1966) mentioned "We can know more than we can tell", some knowledge can be taken for granted but some are difficult. Explicit knowledge is relatively easier to be recorded, pointed to, retrieved and processed, while tacit knowledge is the part that harder to “tell”, harder to transport, receive or quantify (McMahon et al. 2004). Before all other downstream processing, how to capture and how to represent knowledge systematically still remains a challenge. The way to capture and pre-process knowledge can be done through many methods, such as audio and/or video recording, writing down, or interactively annotating. The creation of knowledge typically moves between the two extremes of formal and informal (Staab et al. 2001), for examples from free-style text notes to formal codification, such as XML. These aspects will be described more specifically in the later chapters. In this chapter, the focus is on a high level of abstraction in regard to knowledge content.

3.3.2 Multiple Engineering Viewpoint

Design is a series of activities that innovate and develop a product in order to satisfy the needs of the users and other stakeholders (Blessing and Chakrabarti 2009), where knowledge related to every sphere of human life through out the PLC is involved (Pahl et al. 2007). Therefore, knowledge can be naturally a multifaceted concept with multilayered meanings (Fei 2002). In other words, engineering design is a complex activity that synthesises knowledge from multiple disciplines or viewpoints, in which every aspect has association.

A viewpoint is an encapsulation of partial information in regard to a specific context or interest (Sommerville and Sawyer 1997). It can be security, marketing, or an engineering viewpoint (EV) for example. In each viewpoint, it can further encompass sub-layers. For example, EVs can be seen as various engineering perspectives of seeing the product in regard to specific objective and concerns within an engineering discipline (Davies 2008). It can be, for example, manufacturing, structural analysis, process planning, and each viewpoint may be associated with an different underlying representation (Lee et al. 2001).

In modern engineering practice the structure and form of the artefact are modelled using computer-aided design (CAD) tools. The current paradigm is to use parametric and associative B-rep solid modelling tools for geometry in association with BOM describing assembly-part structures in PLM tools. Desired functions are described in a variety of ways but generally as a list of requirements or using graphical models (Gero and Kannengiesser 2004). The prediction of artefact behaviour is also achieved in a variety of ways – using classical analytical techniques, through empirical relationships, through model or prototype manufacture and test and by numerous computer-based analytical methods. These methods predict the behaviour of the artefact from MEV – structural, thermodynamic, kinematic, cost and so on (Niazi et al. 2006; Cook 2007).

During the PLC, experts from different EVs may participate within a complex system. Integration of the assessment of behaviour into the design process is currently achieved by creating dedicated models of the artefact for the purposes of evaluation, aided either by passing data between tools (e.g. passing geometric data from a CAD tool to a structural analysis tool) or by embedding tools for the assessment of products into CAD systems. For example a CAD suite may have a structural analysis capability incorporated.

To improve the collaboration within one single PLM system or across PLM systems, a range of different views of products, e.g. geometric view, functionality view, manufacturing view and so on must be integrated (Canciglieri and Young 2003). Therefore, information from different viewpoints must be integrated to form the final system specification, so that knowledge from a variety of viewpoints can be propagated and communicated. This raises the concept of MEV.

The nature of MEV as suggested by Fei (2002) – multifaceted concept with multilayered meaning - is a strong support to engineering collaboration. Collaborations between MEV require knowledge transformation across EVs, in which the non-trivial process is limited by the understanding of each other's viewpoints (Bond and Ricci 1992) and often expensive. Therefore, formal knowledge representation of EVs is a critical desideratum to

interoperability between MEV. This includes data structure, and more importantly, the knowledge structure. A sufficient knowledge representation can further support other knowledge management processes.

To address this, many knowledge models have been developed in the last decades. An important model is the function-behaviour-structure (/state) approach (FBS) (Gero 1990; Umeda et al. 1990; Qian and Gero 1996; Gero and Kannengiesser 2004; Colombo et al. 2007). In this model, designs are created in order to achieve particular functions, i.e. design requirements. These are expressed in terms of the desired behaviour of the artefact. Designers propose a structure (i.e. organisation and form of the artefact) in order to achieve this desired behaviour. A variety of approaches is then used to explore the actual behaviour of the structure. The functions, desired behaviour or structure may then be modified iteratively until a satisfactory solution is obtained. In other words, EVs are used to evaluate the behaviour of the artefact from different viewpoints in order to assist with engineering design activities.

Another important knowledge model is the multi-layered ontology architecture first introduced by Guarino (1998) and similar concept also by Jasper and Uschold (1999), and then adapted and evolved (Lee and Suh 2007; Zhu et al. 2009). It manages engineering knowledge into three layered knowledge models: top-level, domain and application ontologies, where knowledge can be contained and propagated across levels and within the same level.

Although nominally integrated approaches for design have been available for some years, in practice in current CAD the different approaches to modelling of function, behaviour and structure are poorly integrated. There is no general mapping from function through behaviour to structure in mechanical engineering. It is expensive and constraining to embed analytical tools in CAD packages and this has only been done for a limited range of tools. Model transfer between tools often involves extensive and expensive manual interaction which is a non-trivial process limited by the understanding of each other's viewpoints (Bond and Ricci 1992). It is difficult to incorporate new tools into CAD systems or to specialise tools to particular application requirements. Therefore, formal representation of knowledge is critical desiderata to interoperability between MEV.

Classifying information/knowledge into EVs can significantly improve the efficiency and accuracy of information retrieval and collaboration across knowledge domains in the sense of precise KR and sharing (Thouvenin et al. 2005; Aubry et al. 2007), for example to understand the different types of holes or the different types of term "flange" in the

domains of telecommunication and mechanical engineering.

In the domain of knowledge management, some other important issues in regard to knowledge management methodologies and its technological support will be described in Chapter 4 to avoid duplication.

3.4 Case Studies

To help understand the concept of EV and some particular problems engineering design currently confronts, two cases in engineering analysis will be described: cost estimation and finite element analysis. Each case is an EV that requires the corresponding expertise, in which knowledge from other EVs are also involved. The cost estimation case study presents an example where the EVs are loosely coupled with CAD systems, and hence the experts tend to build their own models. The second example – FEA represents a case study where CAD systems and FEA are tightly linked together. These studies were selected to illustrate the scope for utilising the proposed framework in this thesis.

3.4.1 Cost Estimation

Cost estimation has been a critical factor that affects design and operational strategies and business decisions for an enterprise, since underestimation causes financial losses and overestimation lead to losses of business and goodwill in the market (Niazi et al. 2006). Quick and effective response to quotations for customer-made products determines the ability to survive in the competitive market (Veeramani and Joshi 1997).

This has been selected as one of the MEV since it has to be considered for every product or service at some stage. As the focus of this research is on the interface of MEV through CAD, this section offers an introduction to the domain, including techniques, knowledge and approaches used in the cost estimation of a product. The aim of this section is to introduce the reader to the information that a subject matter expert in cost estimating may use and in particular how this could be represented within an MEV environment.

3.4.1.1 Cost Estimation Techniques

Niazi et al. (2006) classify product cost estimation (PCE) techniques into qualitative and quantitative. The qualitative cost estimation techniques are based on the previous manufacturing experience in similar products to identify the cost for new product, thus to provide rough estimation in early design stages. The quantitative techniques provide

more accurate cost results based on detailed analysis of a product design, i.e. calculating using analytical functions based on product parameters or summing consumed elementary units throughout the PLC.

Qualitative Techniques

In the category of qualitative techniques, there are mainly two sub-categories: the intuitive and analogical techniques. Intuitive technologies are based on previous experience, while analogical technologies are based on historical product cost data. Each of them also has further divisions, as illustrated in Figure 5.

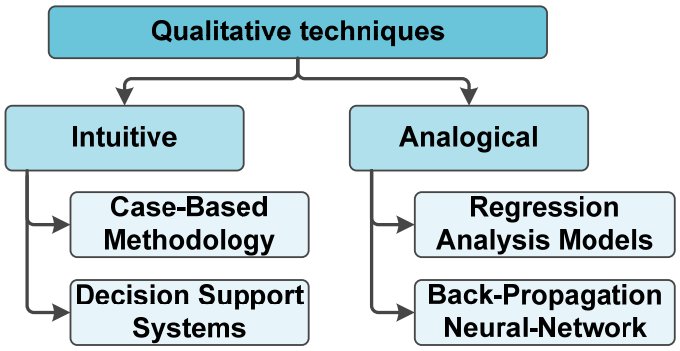


Figure 5 Classifications of Qualitative Technologies (Niazi et al. 2006)

As one type of intuitive qualitative technique, case-based methodology allows quick and early estimation at conceptual design stage by referencing costs in sufficient previous design cases (Duverlie and Castelain 1999). An example of a cost estimating process where the design is similar to a previous design is as follows:

1) Analyze the new design specification.

2) Retrieve the closest design case (source case) from past experiences.

3) Adapt and refine this case by making changes (e.g. assemblies or parts from existing solutions or new ones) according to the new design specification.

4) Repeat the last step until a satisfactory solution is reached (target case).

5) Estimate the cost for the target case by incorporating costs of corresponding changes into the cost of source case.

6) Save the new design solution into the design database as new experience (new case).

Another intuitive approach which is known as decision support systems (DSS), aids decision making on design alternatives through the use of expert knowledge in a domain, such as design and manufacturing constraints including machining processes, machining time and assembly processes.

Apart from intuitive technologies, qualitative techniques also have a category of analogical techniques. These techniques include regression analysis models and back-propagation neural-network (BPNN) models. A regression analysis model is a traditional costing technique based on a linear relationship between known costs of existing design cases and cost drivers; while BPNN aids non-linear cases by using trained knowledge to infer an answer when cost estimation relationship (CER) in regression analysis is not available (Cavalieri et al. 2004).

Quantitative Techniques

Quantitative techniques are categorized into analytical and parametric approaches with further sub-categories, as depicted in Figure 6. Both of them require the detail design or detailed specification to be available, so that costs can be computed based on each elementary unit. For the research presented in this thesis the assumption is that this level of detail is available.

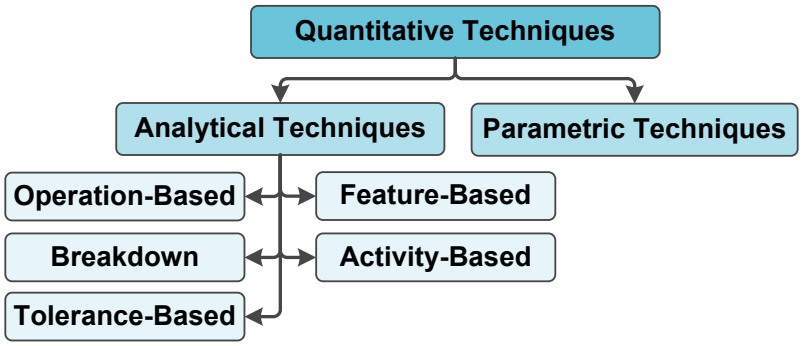


Figure 6 Classifications of Quantitative Technologies (Niazi et al. 2006)

The parametric approach is to calculate costs of a given product by using the knowledge of cost drivers in a number of ways. A cost driver is any factor/parameter that affects the cost of an activity (Blocher 2005). In general, parametric approaches can be used to quantify the unit cost according to for example the production scale, the type of manufacturing processes, the complexity of the product. It can be based on a set of statistical relationships and mathematical formulae that connect the cost of a product or activity to effective cost drivers (Duverlie and Castelain 1999). The advantages of parametric approaches are that they are rapid, accurate, and capable of identifying the

most significant parameters in a product. The accuracy depends on the availability of all required parameters. In some cases, estimators have to estimate the missing parameters, and consequently, accuracy is decreased by including uncertainty. Thus the the greater detail available the better the accuracy can be attained. Moreover, the parametric approach is applicable in all life cycle phases, conceptual, development, production, and operating and support in comparison with the others (Long 2000). Therefore, the parametric approach can be more accurate in the later stages.

Analytical techniques analyze and sum resources consumed by all constituents of a product, in terms of unit cost, operations, activities and so on. As illustrated in Figure 6, it mainly has five sub-categories: operation-based, breakdown, tolerance-based, feature based and activity based approaches.

The operation based approach estimates the time consumed by manufacturing operations, including setup time, operation and non-operation time and so on, when all the required information of a design become available at the final design stage.

The cost breakdown structure (CBS) approach can aid different levels of breakdown. At a product lifecycle level, it aggregates all the costs occurred during the whole PLC. The main tasks in the breakdown structure include design, production, usage, and disposal/recycling (Asiedu and Gu 1998).

The tolerance-based approach allocates the range of variations in terms of costs, in order to aid decision-making processes at early stages before parts and tools are made. Optimal tolerances can be achieved through the trade-off between costs and quality that meet a certain design criterion (Li et al. 2008).

The feature-based approach can be used at various product development stages, such as design and manufacturing. Theoretically, it involves estimating the costs of a product by adding or omitting identified cost-related features, including geometric features, manufacturing features or other features impacting downstream cost of a design (OuYang and Lin 1997; Roy et al. 2001).

The activity-based approach concentrates on the costs occurred by carrying out activities for manufacturing a product or service, e.g. labour cost for machining, tool costs, and so on (Niazi et al. 2006). This approach is applicable at various stages of the PLC (Ozbayrak et al. 2004; H'Mida et al. 2006).

3.4.1.2 Discussion on Product Cost Estimation Techniques

In either qualitative or quantitative cost estimation techniques, cost models need to input cost-related information in regard to the concerned product from various PLC stages, from early design stages (e.g. case-based methodology), embodiment and detail design stages (e.g. parametric techniques), manufacturing (e.g. activity-based costing systems), services, until recycling/disposal (e.g. breakdown structures). The sufficiency of available information (cost drivers) greatly affects the accuracy and applicability of these cost estimation techniques. The reason cost estimating was selected as one of the MEV exemplars is that it demonstrated the need for MEV to obtain a result. For example, a parametric costing tool SEER-MFG (Galorath 2008) needs parameter inputs to calculate cost results. Some parameter inputs are from CAD models – the geometric design viewpoint, some are from other EVS, e.g. manufacturing, materials or other external sources, such as production volume.

Furthermore, retrieving the required data from a complex design determines the efficiency of a costing model. In general practice, cost estimators carry out similar processes as the case based methodology, in which information are interactively collected from customers, markets, various departments throughout the entire organizational structure, such as design, manufacturing, assembly, distribution (Pahl et al. 2007). The results are derived from information which flows further to any interested participants. Thus capturing sufficient information including all required cost drivers from various EVs in a design is critical for the accuracy and efficiency of cost estimation and is largely determined by the responding time for queries by participants and information process. As suggested by Pahl et al. (2007), routine tasks such as variant designs should be largely undertaken by the computer in the future, leaving designers free to concentrate on innovation. Therefore, engineering design processes will largely depend on the effectiveness and efficiency of CAD systems and other engineering tools, which are in turn determined by how well knowledge of MEV can collaborate, including as used in this research the cost engineering viewpoint.

3.4.2 Finite Element Analysis

Finite element analysis (FEA) is used in many application areas, including stress, vibration, thermal, electromagnetic, fluid flow and others (McMahon and Browne 1998). Since the first time Courant raised the concept of FEA in 1943, FEA gradually evolves and the term is coined by Clough (1960). In more recent work, this analysis is named as the Finite Element Method (FEM) by Cook (2007): *"a method of piecewise approximation in*

which the approximating function Ø is formed by connecting simple functions, each defined over a small region (element)", while a finite element is *"a region in space in which a function Ø is interpolated from nodal values of Ø on the boundary of the region in such a way that interelement continuity of Ø tends to be maintained in the assemblage"*.

FEA is suggested as an engineering analysis activity that may encompass six steps in three major stages (McMahon and Browne 1998; Moaveni 1999; Cook 2007):

Pre-processing Stage

1) Mesh the structure or continuum into finite elements.

2) Formulate the properties and equations for each finite element, such as defining allowed nodal loads in a stress analysis or allowed nodal heat fluxes in heat conduction analysis.

3) Assemble elements into a finite element model.

4) Apply boundary conditions, initial conditions, and loading.

Solution Stage

5) Execute a set of linear or nonlinear algebraic equations simultaneously to obtain nodal results, e.g. displacements or temperature.

Post-processing Stage

6) Interpret and analyse the results.

In Step 1) and 4), decision and input data are needed from the analyst to prepare the finite element model. In Step 6), results and derived interpretations may also need to be recorded by the analyst, while all other steps are automatically carried out by computer programs (Cook 2007).

Commercial FEA Software Tools

Many commercial software tools are available for FEA at various scales, however large-size FEA tools dominate the market due to their generality, worldwide distribution,

large user community, better user support, portability and up to date maintenance (Cook 2007). These tools include SIMULIA® by Dassault Systèmes (2011c), NEi Nastran by NEi Software (2011), LS-DYNA by Livermore Software Technology Corporation (2011), LUSAS by Finite Element Analysis Ltd. (Finite Element Analysis Ltd. 2010), COMSOL Multiphysics® by COMSOL (2011), ANSYS® by ANSYS Inc (2011) and many others. Each FEA tool may focus on various applications, viz. structural analysis, thermal etc. FEA tools may have different emphasis, some are complete solution, whilst others focus on pre-processing, or/and solution, or/and post-processing.

Integration of FEA and CAD Systems

In either case, FEA tools work on design objects with geometric representations, which is the same as CAD systems. FEA and CAD were originally developed independently, in which the geometrical construction were fundamentally different (Cottrell et al. 2009). Later on, FEA and CAD systems tended to be integrated to share the supports for both sides (McMahon and Browne 1998; Cottrell et al. 2009). There are generally two alternative collaboration routes between an FEA tool and a CAD system, as depicted in Figure 7 One route can be the CAD system interfaces to an external FEA pre-processor so that CAD system delivers geometry data to it in a neutral format; or in another way, CAD system is integrated with an internal FEA pre-processor, so that its geometric modeller together with this pre-processor interactively prepare a full FEA model for an FEA solver to compute (McMahon and Browne 1998). There can be other variants, either take the advantage of geometric modelling capability from CAD systems or take the merit from FEA systems. For examples, the commercial CAD system Siemens' NX (Siemens PLM Software Inc 2011a) has integrated with an FEA pre-processor, and the FEA tool ANSYS (ANSYS Inc 2011) supports the importation of CAD models.

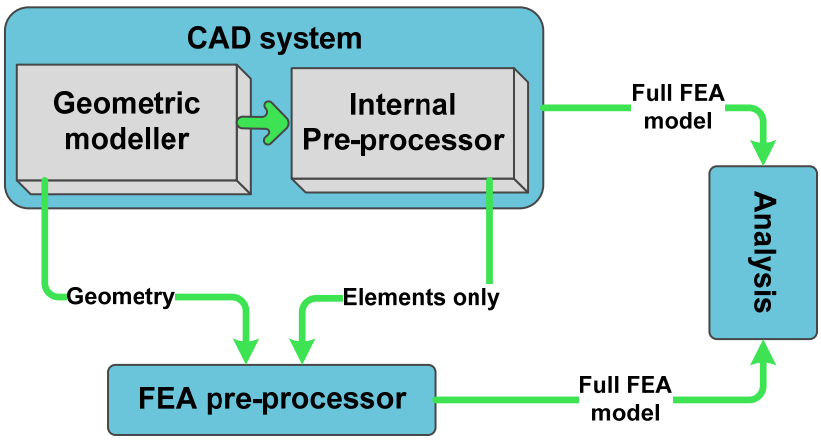


Figure 7 Collaboration Routes between FEA and CAD Systems (McMahon and Browne 1998)

In either route, the fundamental is the exchange of input data to a pre-processor and the association with the geometric models, which are essential inputs for an FEA solver to compute results. In other words, FEA models, as well as their constituents – nodes and elements contains constraints and loads and material properties, all of which can be defined with regard to the external FE geometry (nodes and elements) or the CAD system geometry (faces and edges), and in the latter case then mapped from the CAD system to the nodes and elements of the FE mesh. This mapping is the key to integrating FEA and CAD systems.

However, there still remains issues, for example, even though geometries can be exchanged among FEA systems and CAD systems, e.g. through neutral formats such as STEP (Adams 2006), there are still weakness in exchanging the information (constraints, loads etc.) and the association with geometries/FEA models, in order to minimize the risk of error during the data exchange for model preparation across systems.

3.4.3 Concluding remarks

As a common practice, CAD systems are widely used to create design models and gather engineering knowledge. However, current CAD is dominated by geometry and structure (such as the bill of materials), with little or nothing about product semantics. With regard to managing collected knowledge, the EVs build viewpoint dependent models (such as specific cost models in spreadsheet or FEA models within ANSYS) in association with CAD models or by extracting data from them. Among the EVs, some are tightly integrated such as FEA feature in some CAD systems; some are loosely coupled such as cost estimation, e.g. in spreadsheet models or the SEER software package the users are responsible for duplicating or gathering information.

Apart from the limited KR for semantics, it is even absent for a coherent approach to KR for MEV, which may provide a flexible (comprehensive or simplified) ability to incorporating EVs seamlessly and open the possibility for further processes. From both cases in cost estimation and FEA, it is found that the exchange of data associated with geometries is very critical in engineering design, and the situation remains difficult, especially in terms of sharing knowledge among different EVs, such as using geometric knowledge to support FEA or cost analysis, or to support geometric modelling with the knowledge of engineering analysis. This implies that information created with legacy tools and new tools may need to be imported or supported across different systems, even though some features have been already integrated in many PLM systems. Furthermore, knowledge process automation is also a challenge, for example, to automatically use cost

related knowledge to support decision making during engineering design.

3.5 DRM Models and Concluding remarks

According to the introduction in Chapter 1, an initial reference model can be sketched as Figure 8, where three major challenges in KIM for engineering design are initially drawn out, particularly in the domain of mechanical engineering: knowledge representation while addressing association; knowledge management in addressing MEV to aid engineering process; system extendibility; and information/data interoperability. To address each challenge, there may be many solutions, such as DBMS, file systems, process and Meta modelling technologies as briefly introduced in Chapter 1. However, none of them can systematically cope with all these challenges confronted by KIM. In recent years, the comprehensive support - PLM systems has been actively developed to address all these aspects, in which KIM is the core and deployed varieties of computational enablers. It should be noted that the causal link indicates positive support unless explicitly marked. For example the more translators needed the less efficient the data exchange process is an example of negative impact in Figure 9, while all other links refer to an enhancement.

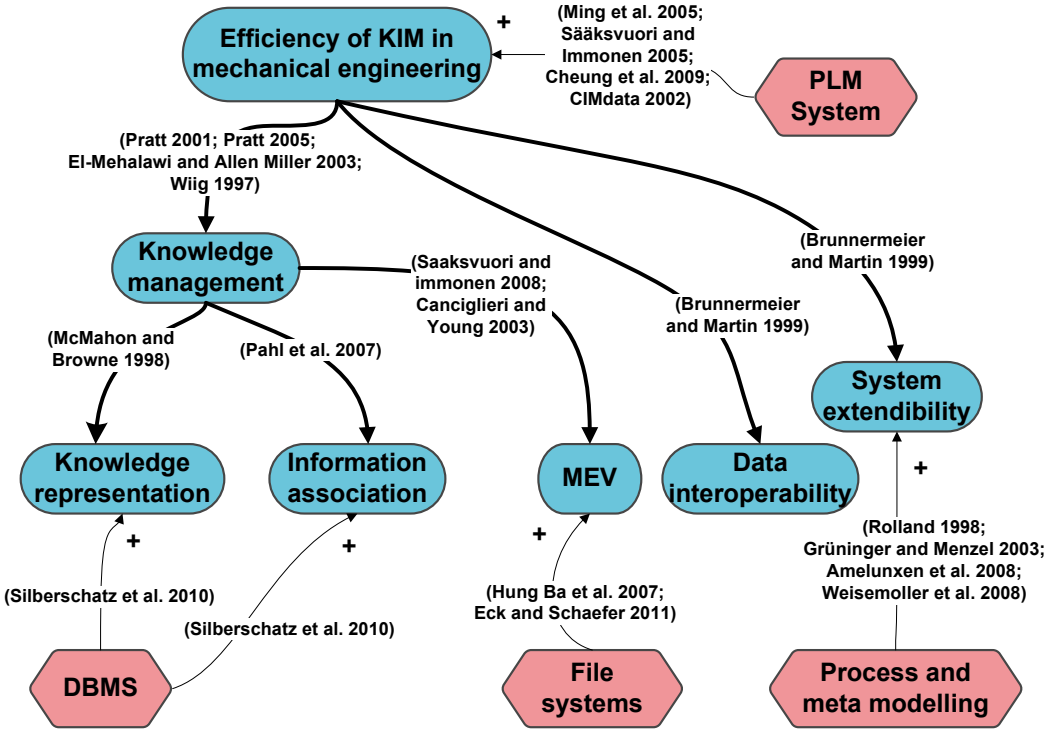


Figure 8 Initial Reference Model with Potential Support

In this chapter, according to the observation on the state of the art in PLM systems, CAD systems and knowledge management in engineering design, an updated reference and impact model can be elicited as Figure 9. According to the project aim, the key factor is to improve the effectiveness and efficiency of knowledge and information management in

the mechanical engineering domain. The success factors can be: whether knowledge can be efficiently acquired and represented; whether association can be maintained among information entities; whether knowledge of diverse expertise can be shared explicitly according to its context; whether data can be explicitly understood therefore to improve interoperability; and whether the system can be extended. Consequently, the measureable success factors can be deduced as marked with dotted border blocks in Figure 9: the automation level of knowledge processes, the level of formality in knowledge representation, whether data can be exchanged efficiently with external systems, the number of translation programs required for data and tool integration, and the resource required in tool downgrading and/or integration.

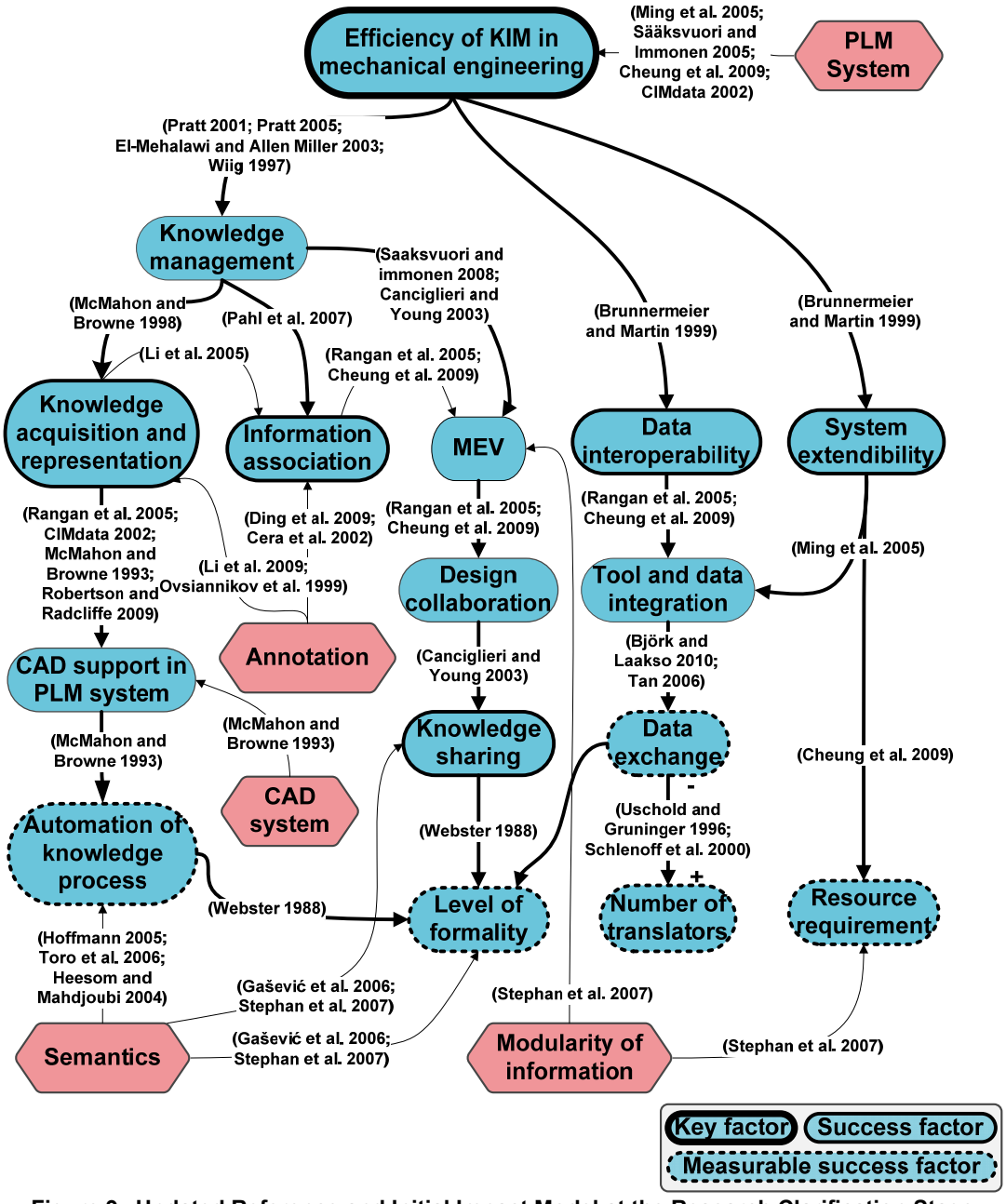


Figure 9 Updated Reference and Initial Impact Model at the Research Clarification Stage

Regarding knowledge management, three challenges are derived: knowledge acquisition and representation, information association and MEV, which are in fact inter-related. There is a need for a coherent approach to semantic KR in addressing MEV, which means a formal and explicit description of knowledge. This will open the possibility for further processes. The automation of such processes, namely the process-ability, is decided by the level of formality in knowledge representation. This is a prerequisite to efficient and effective collaboration among MEV since knowledge from different EVs must be rigorously defined in order to be explicitly interpreted and communicated.

On the other side, data interoperability refers to data integration, and system extendibility refers to tool integration. Both of them imply data sharing and exchange among tools, which is largely affected by the number of translation programs required, and also the standards of data formats. Moreover, the system extendibility is also decided by the resource requirements. For example, in the case that a system is not affordable, whether it can be readily downgraded may be desired. In another situation, it is also a challenge whether the overhead is affordable, including the skills and labour resources for the customization or integrating more services.

According to the literature sources quoted in this chapter and many more in the next chapter, there are some potential solutions available to address each of these challenges, as the potential support depicts in Figure 9. Annotation technologies hold the promise to capture, represent, manage, and process knowledge while maintaining association with other information entities. Semantics are suggested as a key factor to many issues, including knowledge representation, sharing and process automation. The modularity of information (i.e. faceted concept management) may potentially contribute to incorporate engineering knowledge in diverse MEV and system adaptation.

This initial exploration helps to form a diagram for areas of relevance and contribution as shown in Figure 10, in which the relevant technologies are identified and scoped, and the major contributions are clarified. The aim of this research is to assess how engineering design processes can be improved through a general knowledge and information management solution, including establishing product definitions and engineering analysis. This leads to intense investigation on these computational enablers such as annotation and ontologies, and some key related aspects such as knowledge acquisition and representation, knowledge sharing, data exchanging etc, which will be described in the next chapter.

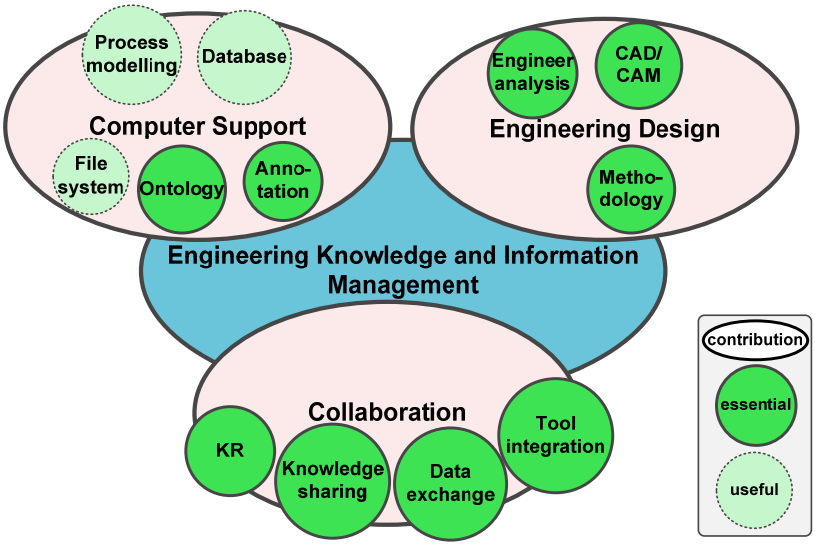


Figure 10 Updated Areas of Relevance and Contribution

Chapter 4 Background – Part II

Previously, it has been found that knowledge and information management (KIM) for CAD systems in the mechanical engineering domain is still a significant challenge. This challenge gives rise to three research questions:

Q1: How can knowledge be captured and represented to aid CAD systems?

Q2: How can knowledge and information interoperate?

Q3: How can engineering services/tools be integrated with CAD systems?

This chapter introduces some important computational enablers that hold the promise to address these challenges in assisting with the engineering design process, including annotation and ontology. According to Lortal et al. (2006), annotations enable communication, and can be experience associated with a document, thus to help people to remember, to clarify, to think and to share (Ovsiannikov et al. 1999). On the other hand, ontology is a "specification of a conceptualization" (Gruber 1993), namely provides the meaning in a knowledge base (Gomez-Perez et al. 2004). The combination of both has the potential to improve information retrieval and interoperability (Uren et al. 2006), thus to overcome the challenges.

Because of past research where annotation and ontology have been highlighted as important enablers, these were selected for use within this research. As the challenge is in the engineering design process and based on reviewing the literature the following three hypotheses were proposed where each one leads to further questions. The following will be tested in this research:

H1: Annotation can be used as a mechanism to capture knowledge and as a medium to represent knowledge while maintaining associations among information entities.

H1-Q1: How can annotation be used to capture knowledge?

H1-Q2: How can annotation be used to represent knowledge?

H1-Q3: How is the association maintained?

H1-Q4: How have annotation technologies been used in the engineering field and what are the weaknesses of current applications?

H2: Ontology can be used as an approach to construct, control, manage and process

semantics so as to aid engineering design by incorporating heterogeneous engineering expertise.

H2-Q1: How can ontologies be used to define semantics?

H2-Q2: How can ontologies be used to manage semantics in respect of incorporating various ranges of expertise?

H2-Q3: How can ontologies be used to process the semantics?

H2-Q4: How have ontological technologies been used in the engineering field and what are the weaknesses of current applications?

H3: The combination of annotation and ontology may provide a solution for KIM, which constructs a platform for CAD systems to incorporate other engineering services/tools.

H3-Q1: How can annotation and ontology respectively aid CAD systems?

H3-Q2: Do annotation and ontology complement each other? If yes, how annotation and ontology can be combined; otherwise are there other alternatives?

If these derived questions can be answered, the hypotheses can be validated. In order to answer these questions and validate these hypotheses, the investigations reported in this thesis were carried out.

In this chapter, the concept of annotation is introduced, together with the associated technologies and their application and the current status in wide fields, with particular attention to mechanical engineering applications. The concept of ontologies and the related technologies are also comprehensively described, such as specification languages, rule languages, modelling methodology, and the state-of-the-art of their application and development.

4.1 Annotation

In many engineering fields, a great deal of development is based on existing information, in particular the re-use of existing data and information, rather than creating new raw data (Buneman et al. 2005). The quantity of such information is ever increasing, as is the engineering workload to process the information. This implies that a systematic and efficient information management system is needed to aid information reuse, while the mechanism to capture and record new information cannot be neglected either.

As previous chapters introduced, there are weaknesses in information integration in PLM systems, customization of PLM systems, and also supporting geographically distributed working environments in order to aid global completion and ever changing market. These challenges all call for the improvement in information technologies for knowledge capturing, representation, searching, exchanging and reuse.

Since information management is critical in a collaborative environment, some technologies have been developed and deployed over time, such as file systems (Hung Ba et al. 2007; Eck and Schaefer 2011), database management systems (DBMS) (Silberschatz et al. 2010), and process and meta modelling (Rolland 1998; Grüninger and Menzel 2003; Amelunxen et al. 2008; Weisemoller et al. 2008), annotation technologies (Ovsiannikov et al. 1999; Kiryakov et al. 2004) and so on. Among these technologies, annotations have been playing an increasingly important role in facilitating the sharing of views and interpretations of information, which hold many advantages over the others due to their nature, from capturing information, storing, retrieval and downstream processing.

However, annotation, as a promising technology, has been underdeveloped in the engineering domain, especially in enriching semantics. In the following section, the generic concepts of annotation will be introduced, and followed by the state-of-the-art of development in annotation technologies. Based on the observation of these technologies and applications, annotations are classified in various ways. At the end of this section, current research gaps in the mechanical engineering field are concluded.

4.1.1 Introduction to Annotation

According to The Oxford Dictionary of English (Soanes and Stevenson 2005a), annotation is defined as “a note by way of explanation or comment added to a text or diagram”. In a digital context, annotation is a piece of data such as a written note, a symbol, a drawing or a multimedia clip that is added to an original information object at a particular location (Ovsiannikov et al. 1999). A generic annotation contains two elements: the *annotation content* and the *annotation anchor*. The annotation content is the data of the additional information that one wishes to add. The annotation anchor is the pointer referencing the address at which the annotation is located (Brush et al. 2001; Wang 2005).

Annotations can be represented in various formats, including but not limited to raw text, XML (Bray et al. 2008), the resource description framework (RDF) (Beckett 2004), and MPEG-7, the Multimedia Content Description Interface from the Moving Picture Experts

Group (Martínez 2004). XML is an extendable mark-up language for transporting and storing data (W3C Schools 2008). RDF (Bray 1998) is a framework for describing Web resources by defining its properties and property values with assigned uniform resource identifiers (URIs). And a URI is an identifier consisting of a sequence of characters to locate Web resources (Berners-Lee et al. 2005). MPEG-7 is a standard representation to encode multimedia data, including video, audio, and 3D objects.

In general, annotations have been used widely. They have become an essential part of people's daily work and are inherently valuable to many information management tools. For example, people may add text notes or drawn diagrams to a written work to record their flow of thinking, to express their opinion, to share information with other participants, and many other purposes (Ovsiannikov et al. 1999; Lortal et al. 2006). People may also annotate photographs, 2D floor plans or 3D geometric models to identify elements of a scene, explain design intent or share information with other design teams. The use of annotations will be further discussed in the later sections.

In the engineering design process, annotations can be used to collect design requirements and designers' opinions at early design stages, enhance the ability of communication, interpret design issues, prevent the loss of information in the PLC, enhance collaboration between various participants by exchanging annotation data (e.g. between different design teams, or between product providers, post-sale service workers and product end users), and also many other purposes. Annotations can also contribute to information management in terms of assisting with storing, retrieving and interpreting information since the more advanced annotation technologies offer semantic features, and support by some computational enablers, e.g. knowledge base and ontologies. Based on current research work, a classification of annotation approaches is presented in the next section together with some example cases and their applications.

4.1.2 The Classifications of Annotation

Annotation has been a well studied topic in recent decades. Many annotation techniques and systems have been proposed and developed in many domains. In this research, existing annotation approaches are classified into six categories with further subdivisions (Figure 11), according to the targeted media, the audience, the rendering system, the usage and function, the representation and the storage location. The studied annotation approaches are explained in more detail in the later sections, compiled with this classification structure.

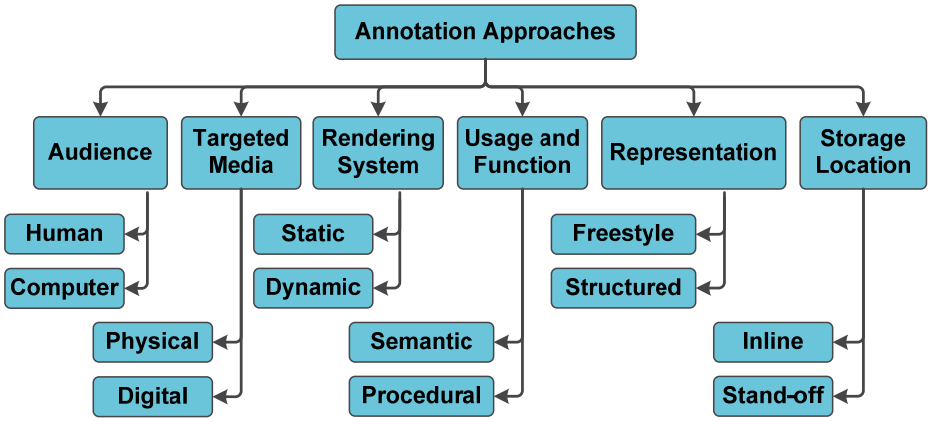


Figure 11 Overview of Classification of Annotation Approaches

Audience

In terms of audience, annotations can be classified into those directed at a *human audience* and those at a *computer audience* (Davies 2008). In the first category, annotations must be human readable, where the annotator can be any participant, either an individual or a team from any type of stakeholder who wish to share information with others. For example, an evaluation team may act as an annotator to add notes as feedback for a specific design team as the audience. In the case of a computer audience, the annotations are fed to one or more computer programs to manipulate the information, e.g. searching through text files, information extraction and so on. In order to be processed and comprehensible by computer programs, annotations must be strictly formalized by complying with a specific syntax or schema and well defined structures (Davies 2008).

Targeted Media

Annotations have been used for hundreds of years, from when monks made notes on the documents they were illustrating to digital annotations using computer programs in the present day. Annotations have been widely used on various target media. Two main classifications - physical and digital - for the target media for annotations, further subdivided as shown in Figure 12, have been identified.

As mentioned above, annotations are often made on paper documents in daily life, and are also made on digital text documents including common word processor and document distribution formats. As Internet bandwidth increases, multimedia documents including audio and video formats with annotations added are able to be communicated. However, 3D objects are differentiated from other types of media due to the complexity of 3D object

representations. The 3D annotations can be applied to a geometric mock-up in physical form or a CAD model in a digital form. Annotations have applied to three-dimensional objects such as 3D maps of terrains (Maass and Döllner 2006), or genome representations (Reeves et al. 2009; Asbury et al. 2010), and many others (Li et al. 2009b). In engineering design, 3D annotations are normally created in the embodiment and detail design stages, while the others may assist with design at any stage, such as using multimedia annotation for design demonstration, and text annotations for evaluation, approval and peer review of documentation.

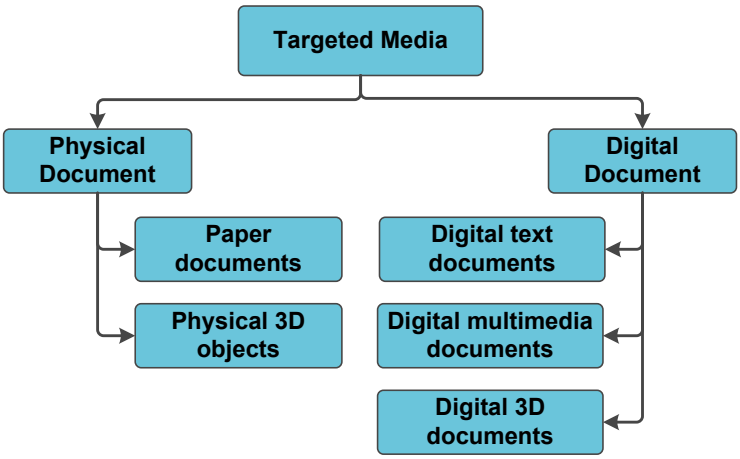


Figure 12 Classifications of Annotation by Target Media

Rendering System

There are many different tools and systems to create and manipulate annotations. Wang (2005) suggests that web-based annotation systems can be classified as proxy-based and browser-based in terms of how annotations are rendered on a request. In the proxy based approach, annotation data are merged with the original document as a new webpage, and committed to a server, which can be returned in its original style. In a browser-based approach, annotation data are saved separately. An enhanced browser retrieves the annotation data and the associated object, and renders them in a desired representation style as requested. In more general terms, the author extended this definition to any digital annotations as *static* and *dynamic* annotations. The Static approach implies that annotations are delivered as saved, and unchangeable (hardcoded); Dynamic refers to adaptable annotation content or/and representation style depending on the specific request.

Usage and Function

The use and the applications of annotations have been various in their history. Marshall (1997), Wang (2005) and Ovsiannikov et al. (1999) summarized the general purposes of using annotations as follows:

- To assist annotators to *remember* what they noted.
- To assist annotators and audiences to do further *thinking*.
- To help annotators and audiences to *clarify* information, such as translating to other language, or interpretation of another viewpoint.
- To contribute to information *sharing* between annotators and audiences.

From the viewpoint of doing further research and developing applications to serve PLM systems, another classification from Davies (2008) considers a higher level of abstraction: *semantic annotation* and *procedural annotation* as described in Table 3, in which semantic annotations are more about to remember, to clarify and to share while procedural annotations are more about to think and to share.

Table 3 Classifications of Annotation by Usage and Function

Sub-category	Description
Semantic annotations	Normally describe the information entity and its constituents. The annotation content is organized by predefined concepts or relationships, making annotations more meaningful and explicit to a model, a domain or within a certain context (Kiryakov et al. 2004).
Procedural annotations	Describe the procedures or processes of manipulating the information and/or its constituents, or provide information to drive that manipulation. Procedural annotations mainly contribute to design process reuse and often work in conjunction with semantic annotations, as it requires more semantic information. For example, descriptions of dimensions, material and material properties for a design part are semantic annotations, while the processes of using such information to compute the mass through a formula are procedural annotations.

Semantic annotation is also known as descriptive annotation, or conceptual annotation (Theodoulis et al. 2003). The primary purpose is to interpret a subject within a certain context to avoid confusion with the meaning in other domains. For example, if “flange” is an instance of a concept in the mechanical engineering domain, it may refer to the rim of railway wheel, but may imply a special type of cable joint that connects electromagnetic

waves in microwave telecommunications. Therefore, richness in semantics enables annotations to provide more information to serve various purposes, which makes MEV representation possible. This contribution is critical to collaboration crossing domains in PLM systems. Well-structured semantic annotations can further assist with automatic processing, such as indexing, information retrieving (IR), and natural language processing (NLP) thus further strengthening the procedural capability (Bonino et al. 2003). In other words, semantic annotations are the foundation of procedural annotations.

Representation

Annotations may also be classified as *freestyle* (informal) or *structured* (formal) according to their representation. Freestyle implies that additional information is arbitrarily created and added to the target without a formal structure, such as random highlighting, underlining, red-lining or short notes. Structured annotations are represented in a pre-defined schema, and managed in a structured way, e.g. indexing or categorizing the annotation data which are described in a formal language, such as XML or RDF (Bonino et al. 2003; Kitamura et al. 2006; Ding et al. 2009). Although freestyle annotation is less maintainable in comparison with structured annotation, it is still widely adopted as it is easy to create and handy for many general purposes. However, the structured annotation approach has enormous advantages, including more straightforward maintenance, semantic richness, self-explaining capability, self-annotation (i.e. annotation to other existing annotations) and process-ability (Wang 2005).

Storage Location

In terms of the approaches to save annotation metadata, annotations can be divided into two classes, *inline* or *stand-off*. In the case of the inline approach, also known as embedded annotation, annotations are embedded into the original objects, i.e. annotations with the target object are saved together. In contrast, annotations can be isolated from the object being annotated and stored as separate files in different locations. The annotation uses its anchor to reference back to the original objects. This is called stand-off annotation, also known as by-reference (Davies 2008) or out-of-line (DeRose 2004).

Along with the increasing complexity and the size of the original design data, also the changing working environment, the stand-off approach presents some major advantages over the inline approach. Inline annotation is limited to the embedding extra information

inside the original object, thus cannot avoid some interferences between annotations (especially when XML is used), and also between annotations and the target object, but a stand-off approach enables the annotator to save the annotation separately, without interfering with any others. This feature solves the overlapping issue, which happens when two or more targeting objects share common portions (DeRose 2004). Also stand-off annotation offers the possibility to annotate bi-directionally (e.g. to either retrieve targeting object or to find the annotation by its target), or even referencing to multiple objects (multi-directional) (Ovsiannikov et al. 1999). However, while stand-off annotations are able to provide many advanced features, they need extra care on maintaining the association – annotation anchors.

4.1.3 The State of the Art in Annotation

In this section, the state-of-the-art of annotation approaches is explored. In order to understand general use of annotation, this review-based study makes observation on many application fields. The annotation applications in engineering design processes are particularly addressed while mapping to the previously mentioned classifications.

Annotation in Specification Formulation

Generally, many current annotation approaches are used by designers to record design intent. One of the most common approaches can be the static inline annotation functions provided by word processing tools (e.g. Microsoft® Office Word (Microsoft Corporation 2009)), including track changes, comments, texts underlining and highlighting. Similarly, Re:Mark (Ovsiannikov et al. 1999) and Adobe® Acrobat® 9 Pro (Adobe Systems Incorporated 2009) offers commenting functions to assist with collaboration through PDF document reviews, such as text notes, redlining[4] images in a PDF file. These approaches fall in the scope of a human audience, as well as computer audience since the digital annotation contents are searchable. They are often used at the early design stage, when more detail design concepts have not yet formed, and designers are still limited to raw information in text documents. Also it is often used in review and approval workflows.

In early design, standard word processing tools are often used in the collaborative development of design specifications. Increasingly, specialist requirement management

[4] Redlining refers to the practice of marking up, often in a red pen, mistakes or changes needed in a drawing or diagram.

tools are also used for these activities (Alatalo et al. 2007), and in these tools annotation is usually an integral part of the capability. Another requirements management tool is called Cradle REQ (3SL 2009), which is able to retrieve engineering project requirements automatically from word documents including the history records, with external user annotations processed by its built-in parser. Furthermore, Setchi et al (Setchi et al. 2011) introduced a tool for semantic-based information retrieval to aid information gathering and idea generation process at early design stage. The tool annotates an image on a web page using significant keywords in the surrounding texts of the image. Annotations are then mapped to ontology, thus the semantics can be semantically queried and information can be retrieved.

Annotation for Communication and Collaboration

Later in the design process, when a geometric design model becomes available, designers and any other participants are able to add or reference annotations to the actual design model for purposes of comments, analysis, review, evaluation and approval. Hisarciklilar and Boujut (2007) addressed the important role semantic annotations can play in collaborative design processes. They addressed two situations of design activities. In an asynchronous situation, design activities are carried out by individual designers, where annotations can be made to record design intent, a list of decisions, remarks, explanations of a CAD model and so on. In a synchronous situation, design evaluation and reviews need to be carried out through a real-time mechanism, such as holding a review meeting. In this case, annotations can be used to formalize the oral discourse, and ensure issues are recorded within a certain context, e.g. a comment can be referenced to a particular part of a 3D model. In the current state, geographically distributed working environments have been an obstacle for design collaboration. The most popular peer-to-peer communication mechanisms cause difficulties of maintenance where communication activities are carried out by individuals and may be not logged or managed centrally. Therefore, Hisarciklilar and Boujut (2007) proposed a scenario that designers can share information for 3D CAD models through a forum-based interface powered by an annotation server.

There is also some similar research with focus on text content on the Web. CritLink (Yee 2002) is used to add and view annotations on web pages. The anchoring mechanism is hypertext linking, in which a bi-directional linking technique is used to allow tracking of both annotation and the object. Another case is called Annotator (Ovsiannikov et al. 1999). Both of them use a static rendering mechanism, and interestingly, Annotator displays

annotations in an index-card style. A key concept is that all communications between client and the Web goes through a proxy server. During annotating, an enhanced browser allows the end users to continually annotate a web page. When the user commits the changes, annotations are extracted from the annotated Web page by a proxy server, and saved into the annotation database with an extended URI while the original Web pages remain unchanged. This sets the user free from needing write permission on the pages. When the reader requests a web page, the proxy server simultaneously retrieves the webpage from the Web and annotations from the database and then merges them as a combined webpage for users to browse.

Another similar stand-off approach is called Annotea (Kahan and Koivunen 2001), in which annotation data is saved in one or more database servers with assigned URIs. On a request for a particular portion of a web page, e.g. text sentences or paragraphs, a plug-in enhanced browser will use a pop-up window to dynamically display the existing annotation for the readers by searching through all involved databases through a proxy server. Based on Annotea, another research group proposed a multimedia equivalent, Vannotea (Schroeter et al. 2006), with substantially more powerful features that applies to wider range of target media. Vannotea provides a collaborative environment allowing participants to discuss, analyse and annotate not only text pages, but also multimedia documents including images, audio and video contents, and also 3D objects in a collaborative way, where annotation content includes participants' discussions, personal notes, anchored to either the whole multimedia file, or a portion of the object simultaneously, e.g. particular frames of a video or audio based on timestamps.

Annotation for the Collection of Specialist Viewpoints

This section has concentrated on the later stages of design processes where detail design models are available. In current engineering design, CAD systems have been widely adopted for decades, and become an essential aid to any product design, especially in the field of mechanical engineering. How to efficiently edit a CAD design model and to reuse design parts are vital questions for industrial application.

Davies (2008) proposed a hybrid annotation framework for both semantic and procedural annotations, which is able to annotate 3D CAD models for multiple viewpoints throughout the PLC, including the manufacturing viewpoint, evaluation viewpoints and so on. This approach is to adding semantically rich information from various viewpoints to features (e.g. simple thru-holes, fixing holes, balance weights etc) of a 3D CAD model. For

example, designers can add annotations with manufacture and stress analysis viewpoints to the CAD model. A manufacturing engineer can determine a hole as a simple thru-hole feature learnt from the annotation information and identify that it requires machining operations such as drilling. Information about these operations can also be added by annotation. At the same time, an analyst with a stress analysis viewpoint who has no interest in manufacturing information, but would find a useful indication that the hole is a fixing hole, which implies some boundary condition information for the analysis. Apart from the multiple viewpoint issue, this framework also supports procedural annotation to help with design process reuse. This is done by manipulating the annotations to control the operations. In the previous example, a CAD model designer may reverse the design process in order to remove the hole, or the manufacturer is able to define the appropriate manufacturing instructions to produce a prototype according to the knowledge of drilled thru-hole. The important contribution of this research work is the ability to collect multiple viewpoints for various types of participant in a collaborative working environment and the design process reuse.

MATRICS© (Managing Annotation for Training in an Immersive Collaborative System) (Thouvenin et al. 2005; Aubry et al. 2007) is another semantic annotation approach to collect specialist viewpoints that aids CAD models or digital virtual 3D mock up based on CAD/CAM design tools. The annotation anchors are defined by 3D coordinates of the mock-up object, and annotation content represented in a structured form. The associated knowledge-based system maintains and maps annotations into three concepts (viewpoints): the mechanical design concept including materials, scientific and technical domains; geometrical description of the mock-up; and the method related concept, namely, the methods in field expertise for engineering design. Queries are made on a selected concept, and the ontology-powered knowledge management system filters the annotation pool accordingly, and retrieves the best relevant results. Experimental work was carried out to test the effectiveness by comparing two groups of students: Group A answered a list of test questions with the assistance of MATRICS©, while Group B answered questions without it. The results showed that 69% of correct answers are achieved by Group A, against 27% made by Group B.

A hybrid of the semantic and procedural approaches – Funnotation (Kitamura et al. 2006) – is designed to aid CAM design. Its core module is an ontology, which is based on a device-centred viewpoint and “role” concepts that represent the functions of a device. For instance, a tool having sharp toothed edge (class constraint for role) can play a “cutting” role (role concept) in a “manufacturing” relation (role context), and is called a “saw” (role

holder). Annotations are obtained by processing semantic web documents, such as a description document of a machine or a summary report of a component. This complete semantic annotation model represented in RDF contains four elements: the function of a device; the way the function is achieved; the functional decomposition structure of the device; and alternative solutions to achieve the functions. When a database is filled with sufficient annotation entries, given a functional design specification, the system opens up the possibility of automatically searching for suitable parts that have the required functions and generating assembly processes to use the potential parts.

Instead of dealing with text documents, the 3D Annotation Framework (3DAF) (Bilasco et al. 2006) is a semantic procedural annotation that assists with design reuse by automatically manipulating 3D geometries. Annotation databases evolve through an annotation manager by adding, removing and updating semantic profiles of 3D scenes in various source formats, such as VRML or its successor X3D[5]. Users send semantic requests to the annotation repository through the query manager to retrieve the desired semantic information pointing to corresponding 3D fragments. A fragment integration component of the 3DAF system translates all fragments into a united format (i.e. X3D), and reassembles them into a new 3D model according to the geometric topology defined by the semantic request.

Other than directly manipulating 3D geometries through CAD/CAM tools, there are also some other cases of annotation. A stand-off 3D annotation approach for architectural design called the Space Pen Java applet (Jung et al. 2002) has been developed. It is a web-based system, in which 3D models in VRML format can be simultaneously accessed from a server for users to create freehand textual comments and drawing on the 3D model. If a gesture is made with the recognizable command, the corresponding digital texts or drawings will be generated and saved externally onto the server. Rather than dealing with a digital virtual environment, ModelCraft (Song et al. 2006) is to annotate physical geometries for engineering design, in which digital pen and Anoto paper technologies are used. Anoto paper (Living Paper 2007) is ordinary paper but pre-printed with dot patterns that can be recognized by specifically designed digital pen. A digital 3D model created by a CAD system is printed out onto Anoto paper in an unfolded style. Freehand texts and drawings on the refolded model can be created and captured by the digital pen. All these captured annotations can be merged to the original digital 3D model,

[5] X3D is a file format and run-time architecture to represent and communicate 3D scenes and objects using XML. It is the successor of VRML. (Web3D Consortium 2008)

thus physical drawings are transferred back to the digital world.

Summary of the Observations

A matrix summary of existing annotation approaches based on the classifications is illustrated in Table 4 and Table 5, including some approaches not described above due to space constraints. In these two tables, ✓ indicates YES (approach matches onto the category), × refers to NOT PRESENTED, KB refers to the presence of a knowledge base, and N/S indicates that the author(s) did not specify. Through this research work, we found that annotation approaches and their applications have various purposes. In general, the more complex and advanced the annotation approach is, the more categories it may fall into. Most annotation approaches in these classifications are not mutually exclusive; in fact, many of them are hybrid systems.

Table 4 Classification Matrix based on audiences, representation and rendering system

Approach	Audience		Representation		Rendering System	
	Human	Computer	Static	Dynamic	Free-style	Structured
Annotea (Kahan and Koivunen 2001)	✓	×	×	✓	×	✓
Vannotea (Schroeter et al. 2006)	✓	×	×	✓	×	✓
Annotator (Ovsiannikov et al. 1999)	✓	×	✓	×	✓	✓
CritLink (Yee 2002)	✓	×	✓	×	✓	✓
XIRAF (Alink 2005)	✓	×	×	×	✓	×
DOSE (Bonino et al. 2003)	✓	✓	×	✓	×	✓
Magpie (Dzbor et al. 2004)	✓	✓	×	✓	✓	✓
MnM (Vargas-Vera et al. 2002)	✓	✓	×	✓	×	✓
S-CREAM (Handschuh et al. 2002)	✓	✓	×	✓	×	✓
KIM (Kiryakov et al. 2004)	×	✓	×	✓	×	✓
Shin et al. (2006)	✓	×	×	✓	✓	✓
Funnotation (Kitamura et al. 2006)	×	✓	×	✓	×	✓
Li et al (2005)	×	✓	×	✓	×	✓
Soo et al. (2003)	×	✓	×	✓	×	✓
Davies (2008)	✓	✓	×	✓	×	✓
LIMMA (Ding et al. 2009)	✓	✓	×	✓	✓	✓
MATRICS© (Thouvenin et al. 2005; Aubry et al. 2007)	✓	✓	×	✓	✓	×
Space Pen (Jung et al. 2002)	✓	×	×	×	✓	×
3DSEAM (Bilasco et al. 2005)	✓	✓	×	✓	×	✓
3DAF (Bilasco et al. 2006)	✓	✓	×	✓	×	✓
Pittarello and Faveri (2006)	×	✓	✓	×	×	✓

Table 5 Classification Matrix based on usage and function, location, and targeted media

Approach	Storage Location		Usage and Function		Targeted Media
	Inline	Stand-off	Semantic	Procedural	
Annotea (Kahan and Koivunen 2001)	×	✓	×	×	Web
Vannotea (Schroeter et al. 2006)	×	✓	×	×	Web, Multimedia & 3D
Annotator (Ovsiannikov et al. 1999)	×	✓	×	×	Web
CritLink (Yee 2002)	×	✓	×	×	Web
XIRAF (Alink 2005)	×	✓	×	×	Raw data
DOSE (Bonino et al. 2003)	×	✓	✓+KB	×	Web
Magpie (Dzbor et al. 2004)	×	✓	✓+KB	✓+KB	Web
MnM (Vargas-Vera et al. 2002)	×	✓	✓+KB	✓+KB	Web
S-CREAM (Handschuh et al. 2002)	✓	✓	✓+KB	✓+KB	Web
KIM (Kiryakov et al. 2004)	×	✓	✓+KB	×	Web
Shin et al. (2006)	×	✓	✓+KB	×	Web
Funnotation (Kitamura et al. 2006)	×	✓	✓+KB	✓+KB	Web (CAM)
Li et al. (2005)	✓	×	✓+KB	×	Web (CAM)
Soo et al. (2003)	×	✓	✓+KB	×	2D images
Davies (2008)	×	✓	✓	✓	3D
LIMMA (Ding et al. 2009)	×	✓	✓	✓	3D
MATRICS© (Thouvenin et al. 2005; Aubry et al. 2007)	N/S	N/S	✓+KB	×	3D
Space Pen (Jung et al. 2002)	✓	×	×	×	3D
3DSEAM (Bilasco et al. 2005)	×	✓	✓+KB	✓+KB	3D
3DAF (Bilasco et al. 2006)	×	✓	✓+KB	✓+KB	3D
Pittarello, and Faveri (2006)	✓	✓	✓+KB	×	3D

4.1.4 Challenges in Annotation

Based on the exploration on current development of annotation technologies, some issues can be identified in the context of engineering design. And these will be described by emphasizing three aspects: annotation anchor, annotation content and the annotation system as a whole.

Annotation Anchor

For the anchoring mechanisms, effectively identifying the target part of an object, and

persistently maintaining the anchors are the current main issues. Anchoring in text documents is relatively mature. For example, XML information retrieval approach to digital forensics (XIRAF) (Alink 2005) assists with digital forensic investigation on raw data, where data are static and changes are prohibited. Annotation anchors are applied by defining the region of the object with the start and end byte addresses of the raw data. In another approach, dealing with dynamic documents, Annotator applies the anchor by using the techniques of so called atoms and clumps (Ovsiannikov et al. 1999). An atom is the smallest text unit (e.g. a word) or a freehand or square selection of a picture. A clump is a series of atoms that reference to the same annotation. On the other side, the annotation database logs a portion of the Atoms as a memorable sub-string together with the annotation. This sub-string will be used to re-locate the anchor when merging the retrieved annotations with a Web page to answer a request. Wang (2005) also proposed advanced anchoring mechanisms to tackle the annotation persistence issue aiding text documents, including the meta-structure information match, the keyword match, and content semantics match.

The presented research in this theis places its focus on anchoring 3D CAD models, as they are vital in the embodiment and detail design stages. The 3D semantics annotation model (3DSEAM) (Bilasco et al. 2005) presents a generic 3D anchoring mechanism that partitions and localizes 3D objects in order to index their constituents, so that information can be anchored to desired parts of 3D objects. Interestingly, an extended version of MPEG-7 is adapted to achieve this. MPEG-7 provides two types of localization: the structural and the spatial. The structural locator helps to select the content units structurally, and the spatial one helps to identify the targeting portion geometrically. For instance, the 3D object can be a virtual 3D scene illustrating a table or simply a stack of papers. The table can be structurally described as an assembly of rectangular boxes in X3D indicating one table surface and four legs. In another case, a stack of papers on this table is represented as a single box, in which the latest publications are at the top, and draft papers at the bottom. The latest publications can be identified by structurally selecting the box of the papers, and then geometrically referencing the top portion. By localising the 3D object, annotation content can be referenced to the specifically identified elements. However, current annotation systems do not completely tackle the persistent anchoring issue yet for 3D objects, for example consistently updating anchors as the models change or remapping anchors across different CAD systems.

Annotation Content

In the annotation content aspect, the representation, retrieval, use and interpretation of annotations raises some issues. With regard to the representation and use, whether or not annotation data can be recognized as application/platform independent is vital to collaboration among enterprises, and among design teams. Cheung and Schaefer (2010) suggested that XML was the most recognized format according to their study, as where thirty seven out of forty five PLM systems they examined supported XML. A typical feature of XML is that it is naturally extendable, and is designed to define other description languages, rather than being a language itself. Furthermore, some other major standards tend to be compatible with XML, including STEP, RDF, and MPEG-7. This enables the derived or compatible languages to inherit the merits from XML. Thus, the unification of representation offers greater collaboration and compatibility among organizations and systems in data exchanging or design cooperation. Although XML syntax and schema can be potentially used to describe general annotation metadata, consensual terms to define annotation data in various engineering disciplines does not exist yet, i.e. a well recognized way to represent anchors, to attach structured content. Therefore, a universal representation of annotations is needed to solve the issue of application/platform independency.

Regarding retrieval and interpretation, how efficiently annotations can be retrieved and properly interpreted are the key issues. Ontologies are considered to effectively contribute to cluster domain knowledge in aid of searching and managing knowledge. However, manually mapping semantic annotations against an ontology requires expertise, and automation of this process still largely depends on the capability such as natural language processing. More details on ontological technologies will be introduced in Section 4.2.

Annotation System

A sound annotation system should address the aforementioned issues, e.g. consistent and precise anchoring mechanism, and support data exchange internally and externally. Also, another challenge is sharing annotations in a distributed environment. This concerns issues including information security, annotations in concurrent collaboration, data interoperability. Annotation technologies along are still weak in systematically overcome these issues. As noted, the knowledge-based systems (Alavi and Leidner 2001; Verhagen et al. 2011) can be one of the options. One of its computational enablers – ontology will be described in later sections.

Experimental Work

The author also carried out a preliminary experimental work to observe the standard annotation function provided by a commercial CAD system – NX6 (Siemens PLM Software Inc 2011a). This experimental work aimed to ascertain the state of the provided annotation functionalities, in which experimental questions (EQ) are listed and answered as following.

EQ1: How general purpose annotation can be added in NX6?

NX6 supports two types of annotations (or attributes) to record non-geometric information: system and user-defined. System attributes are those assigned by users and recognisable by the system, for example, a name labelled on an object or object attributes to attach numeric values or texts as shown in Figure 13. User-defined attributes are customised and have no meaning to the system. For example, attributes associated to a group of geometric objects so they appear correctly in the bill of material for the product. This feature includes a standard-compliant annotation scheme – PMI with standard ASME Y14.41, and user defined PMI. These standard functions will be referred as NX6 native annotation functions in the rest of this dissertation.

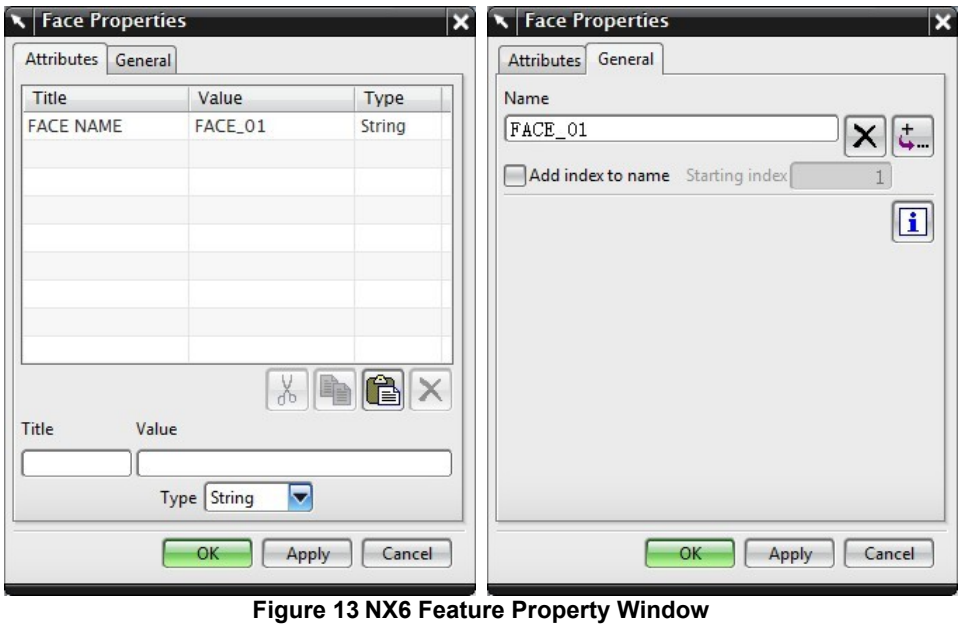


Figure 13 NX6 Feature Property Window

EQ2: What granularity level of annotation can be achieved by NX6?

The NX native annotation function shown in Figure 13 can be applied to any level of the CAD model as NX6 is capable of. This includes the body (i.e. part), face, edge, point, feature and so on.

EQ3: How is general purpose annotation data maintained?

This type of annotation data does not violate geometry, but is still embedded in the same file for the design model. Each annotation entry is independent from the others, without awareness of inter-relations.

EQ4: How general purpose annotation data can be processed?

This NX native annotation function offers limited capability in processing the annotation data. It can be added, cut, copied, pasted, and deleted. Annotations can also be reported or displayed, for example, if an inquired geometric object has associated with a PMI object, however the content is not searchable, neither programmatically through its application programming interface (API).

EQ5: Whether general purpose annotation data can be exchanged?

The experimental observations on this question can be divided into three cases. In the first case, the geometric model was tried to be exported from NX format into an external format (VRML). All user annotations (e.g. object names and attributes) were lost in VRML file. However, PMI annotations can be exported to Siemens JT format but inline with geometry definition. In the second case, the object names of geometry elements can be exported inline with geometry definition through STEP formats. Unfortunately, the other attributes such as PMI are still lost.

The last case is based on an assumption that annotation data is stored in a stand-off repository, which is independent from CAD system but associated with the object names of geometry elements as annotation anchors (Figure 13). This is to explore whether it is possible to maintain a persistent stand-off anchoring mechanism. A NX6 add-on program called the annotation transfer agent was programmed in C and C++ by using the NX API – NX OPEN. The process is as depicted in Figure 14. The NX6 part is automatically assigned names for all faces and then exported from NX format to X3D format through an intermediate format VRML, as it is not directly exportable. Each face described in this X3D file is compared with the original NX6 part, if any face is matched then it will be relabelled as it was in NX6 part. This is to explore whether annotation anchors can be restored if annotation data is exportable and stand-off from geometry definition.

According to the results of the experimental work, the object names of geometry elements (the NX6 native annotations) can be programmatically exported to other formats, and it also can be exported via certain format – STEP. This implies that geometry elements may

potentially provide a basis for robust anchoring mechanism. However, whether annotation data can be exchanged is based on the assumption that other NX6 native annotation data can be stored in an accessible stand-off repository so that supports data exchange. Unfortunately, the NX native annotation data such as PMI is only exchangeable inline with certain format – Siemens JT. In a nutshell, except EQ2, all other experimental questions are only partially satisfied.

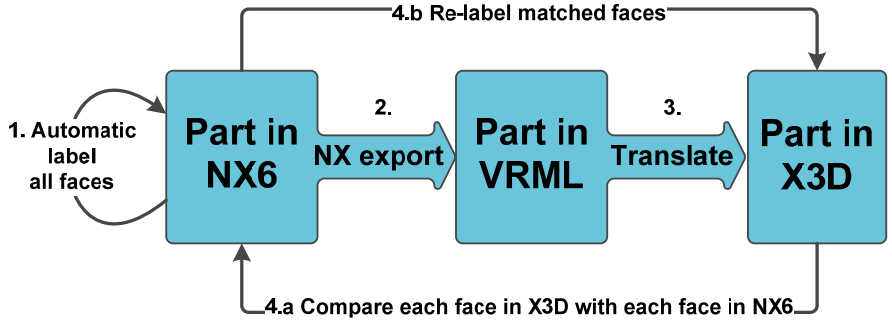


Figure 14 NX6 Annotation Transfer Agent

4.1.5 Concluding remarks

Annotation has been widely studied and deployed for many years in many fields, such as education, computing, biochemistry, engineering and so on. In this section, over thirty existing annotation approaches were observed and classified them into six categories. Based on this study, the research questions in regard to the hypothesis H1 can be partially answered:

H1-Q1: How can annotation be used to capture knowledge?

> Data can be captured as annotations during the annotating process, and can be applied to all types of information object, including text documents, multimedia documents and 3D CAD models.

H1-Q2: How can annotation be used to represent knowledge?

> Data can be represented in the data structure: the annotation anchor and annotation content. The anchor varies depending on the target media, such as text or 3D documents. The annotation content can be in the form of texts, multimedia, or digital 3D objects, either in freestyle or structured way and specified in various languages, such XML.

H1-Q3: How is the association maintained?

Since annotation consists of anchor and content, the association between target object and annotation can be referenced to each other. Annotation data can be stored either stand-off or embedded, which offers flexibilities for maintenance.

H1-Q4: How have annotation technologies been used in engineering fields and what are the weaknesses of current applications?

Although annotation technologies have been applied in many engineering fields, there remain challenges, including the consistency and granularity of anchoring mechanisms for CAD systems. With regard to granularities for annotating CAD models, most current annotation approaches focus on the level of design parts only, although it has potential to be applied to all levels.

Secondly, strictly speaking annotations largely represent information/data, rather than knowledge, since the anchor-and-content structure is too simple to hold semantics for knowledge. The semantics reflect to the knowledge management in addressing MEV. As a consequence, it is incapable of more advanced information retrieval, thus hinders knowledge exchange and reuse.

According to the potentials and gaps in current annotation technologies, the support from annotations can be concluded as the impact model (Figure 15). It is an inherent valuable part of PLM systems, and we argue that annotation can better serve knowledge representation, communication, collaboration and automation and covers the entire PLC, with some improvement, especially ontologies, which will be described in the next section.

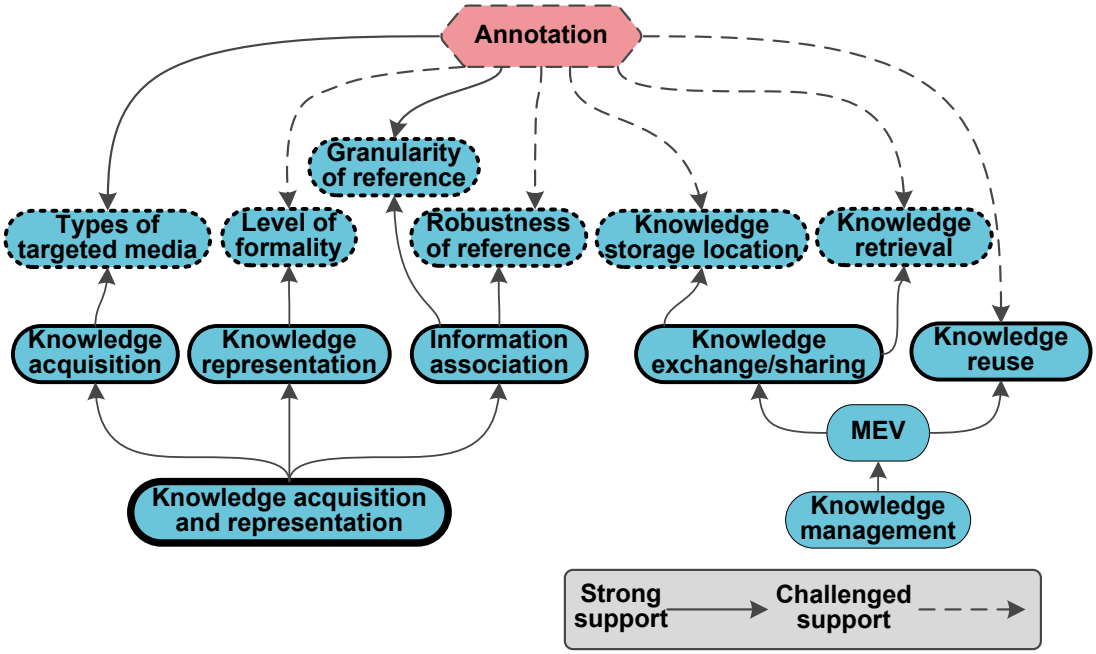


Figure 15 Impact Model for the Support of Annotation Technologies

4.2 Ontology

Ontology was originally derived from Greek words, standing for the study of being, and refers to the subject of existence in philosophy. Ontology is adopted by many computing disciplines over years, especially in AI. One concise and the most widely quoted definition of ontology was given by Gruber (1995), as "an explicit specification of a conceptualization", where the conceptualization is an abstract view of the world, the specification is a formal description to describe the "world". The "world" (conceptualization) means some phenomenon in the world, or some topics, which are established by filling with concepts, objects, and entities in an area of interest, and the relationships that exist among them.

Another formal definition given by Sowa (1999) is as follows:

> *The subject of ontology is the study of the categories of things that exist or may exist in some domain. The product of such a study, called an ontology, is a catalog of the types of things that are assumed to exist in a domain of interest **D** from the perspective of a person who uses a language **L** for the purpose of talking about **D**. The types in the ontology represent the predicates, word senses, or concept and relation types of the language **L** when used to discuss topics in the domain **D**.*

Since AI is adopted in web services, Hendler (2001) and Devedžić (2006) defined a more practical definition for an ontology as a set of knowledge terms that consist of the vocabulary, the taxonomy, the semantic interconnections, some rules of inference and logic for a particular topic.

Ontologies have attracted much attention in the past years in many fields, such as computer file systems (Hung Ba et al. 2007), engineering design (Lim et al. 2009), engineering analysis (Grosse et al. 2005), biology and bioinformatics (Shaban-Nejad et al. 2005; Lambrix et al. 2007).

The most active field of its application and development of ontologies is the Semantic Web (SW) (Devedžić 2006; Gašević et al. 2006b; Bratt 2007; Joo and Lee 2009). Although the Web is an immense source of information, it is not intelligent enough to use information more efficiently since it is originally designed for human audience to read rather than for machines to understand. The Web needs a new ability to explicitly represent knowledge in order to allow web-based applications to semantically understand and use the content in the web pages, such as semantic search engines or information brokers (Devedžić 2006). This is why there is a need for the SW.

In particular, semantic annotation as one of the major trends has been applied in the SW (Uren et al. 2006) to enable web content to be understandable by machines, to enable advanced knowledge representation and management (Gašević et al. 2006b). In the SW context, semantic annotation is more about semantic markup, which is used as a web language and powered by one of the most critical computational enablers – ontology. In the SW, ontologies are used to formally and explicitly establish a knowledge skeleton, in which all data/information are filled, thus computer applications can further process the knowledge (Devedžić 2006). Based on the semantics, ontologies have the ability to interpret, to reason, and to evolve (Ding et al. 2007). The explicit interpretation of knowledge by computer applications such as the previously noted annotation tool Magpie, improves the collaboration and interoperation across knowledge domains, therefore we consider ontology also a desideratum to serve mechanical engineering, where ontology-driven semantic annotation can analogously represent knowledge systematically in engineering context.

Since ontology is used for formal knowledge representation, it requires formal specification languages, modelling tools and methodologies, which will be introduced in the following section.

4.2.1 Ontology Specification Languages

In practice, an ontology is normally modelled in a formal specification language, so that the knowledge can be machine-readable. In recent years, many ontology specification languages have been developed. Corcho and Gomez-Perez (2000) classified them into traditional, web standards and recommendations by the World Wide Web Consortium (W3C) (W3C 2009), and other web-based ontology specification languages, as shown in Table 6, Table 7 and Table 8 respectively.

Corcho and Gomez-Perez (2000) made a comparison among ontology languages based on two main aspects: domain knowledge and inference mechanisms. Domain knowledge describes the static information and knowledge objects in a domain of interest, including concepts, relations, functions, axioms and instances are compared. The inference mechanisms determine how the static information and knowledge can be used for reasoning, deriving dynamic information and knowledge. The authors claimed that the language LOOM® (succeeded by PowerLoom®) is the only one that covers most features in every compared features of domain knowledge. They also claimed that OCML is as good as LOOM in supporting domain knowledge and concepts, and they are also the only languages supporting procedural rules (production rules). But later on OWL

succeeded DAML+OIL, caught up quickly and became a web recommendation, e.g. it does support instances now, and does support first-order logic. With regard to inference and reasoning capability, F-logic and OWL are the most sound and complete languages.

Table 6 Traditional ontology specification languages

Categories	Ontology language	Description
Traditional ontology specification language	Ontolingua (Farquhar et al. 1997)	It provides forms for defining classes, relations, functions, objects, and theories. The syntax and semantics are based on a first-order predicate calculus known as the knowledge interchange format (KIF).
	OKBC (Chaudhri et al. 1998)	Open knowledge base connectivity (OKBC) is a successor of generic frame protocol (GFP). It specifies a protocol that supports an object-oriented knowledge model and a set of operations based on this model.
	OCML (Enrico 1998)	Operational conceptual modelling language (OCML) is a frame-based language that provides mechanisms for expressing items such as relations, functions, rules (with backward and forward chaining), classes and instances.
	F-logic (Kifer et al. 1995)	Frame logic (F-logic) integrates frame-based languages and first-order predicate calculus. It defines ontologies in a declarative fashion. Features include object identity, complex objects, inheritance, polymorphism, query methods, encapsulation.
	PowerLoom® (Chalupsky et al. 2006)	PowerLoom®, the successor to the Loom® (MacGregor 1991), is a variant of KIF. It supports reasoning over the knowledge declared with definitions, rules, facts, and default rules. The reasoning engine - classifier is able to determine relationships between two knowledge descriptions, or deduct a new one.

Table 7 Web standards and recommendations

Categories	Ontology language	Description
Web standards and recommendations	XML (Bray et al. 2008)	XML is not an ontology language, but is able to define a syntax for an ontology specification language or used for ontology exchange. Any ontology language based on XML can inherit its advantages and limitations.
	RDF (Manola and Miller 2004)	Resource description framework (RDF) is in fact a complementary to XML, which enriches the semantics for XML-based data through describing web resources by defining property types, properties and associated values.
	RDFS (Brickley and Guha 2004)	RDF schema (RDFS) is extension to RDF. It defines RDF vocabularies for application specific classes and properties.
	OWL (Smith et al. 2004)	Web ontology language (OWL) is a successor of DAML+OIL (Connolly et al. 2001). It is a language for defining and instantiating Web ontologies, which includes descriptions of classes, properties and their instances. OWL goes beyond XML, RDF and RDFS in the sense of adding more vocabulary for describing properties and classes.

Table 8 Web-based ontology specification languages

Categories	Ontology language	Description
Web-based ontology specification language	XOL (Karp et al. 1999)	XML-based ontology exchange language (XOL) is an intermediate language to exchange ontology definitions among interested parties, whose syntax is based on XML and the semantics is based on OKBC.
	SHOE (Luke and Heflin 2000)	Simple HTML[6] ontology extension (SHOE) is designed to aid user-agents improving search mechanism and semantic information gathering for web documents. Ontology in SHOE is an ISA (i.e. "is a") hierarchy of classes, plus a set of atomic relations between these classes, and a set of inferential rules in the form of simplified horn clauses. Classes inherit relations from parent categories.

4.2.2 Ontology Modelling Tools

There are mainly two categories of ontology modelling tools as suggested by Corcho et al.

[6] Hyper text markup language (HTML) is the lingua franca for publishing hypertext on the World Wide Web. (Hickson 2011)

(2006): language specific tools or language independent tools. Language specific tools are ontology modelling tools that are explicitly developed for specific languages in order to establish corresponding knowledge models, some examples are shown in Table 9. On the other hand, language independent tools have extendable system architecture and allow knowledge model to be established that are independent of any specific language, but generally with support for a set of languages; some examples are shown in Table 10.

Table 9 Language Specific Ontology Modelling Tools

Tool	Ontology Specification Languages
Ontolingua (Stanford University 2005)	Ontolingua and KIF
Loom OntoSaurus (Information Sciences Institute 1998)	Loom®/PowerLoom®
SWOOP (Mindswap 2005)	OWL
KAON (FZI WIM and AIFB LS3 2011)	RDF

Table 10 Language Independent Ontology Modelling Tools

Tool	Ontology Specification Languages
Protégé (Stanford Center for Biomedical Informatics Research 2011)	OKBC, RDF(S), OWL, XML Schema, and more.
WebODE (Arpírez et al. 2001)	XML, RDF(S), OIL, DAML+OIL, OWL, F-Logic, and more.
OntoStudio® (ontoprise GmbH 2009)	F-Logic, RDF(s), OWL, UML, database patterns (e.g. MS-SQL) and more.
KAON2 (Motik 2011)	OWL-DL, SWRL, and F-Logic.

Discussion on Ontology Specification Languages and Tools

To choose an appropriate ontology specification language is largely a decision based on the trade-off between the degree of expressiveness and the inference engine of a language. Considering the expressiveness of OWL, compatibility (based on XML), the large scale of supporting community and its reasoning facilities (supported by some commercial and open sourced engines), the OWL – a W3C recommendation is therefore considered as the most appropriate KR language. Furthermore, another credit of OWL is that it has complementary rule language for richer semantics and a query language that will be introduced in the following sections.

Language independent tools are mainly considered in the present work, as they support more ontology specification languages, thus have better compatibility to exchange knowledge models in other languages. The comparison is mainly made between Protégé

and OntoStudio®, since they are the most actively developed editors and support OWL. Although the performance of OntoStudio® is more stable in comparison with Protégé, it is a commercial toolkit. In contrast, Protégé is open-source application, thus widely supported by global researchers from various fields and the performance also catches up quickly. Besides, Protégé is a modularized tool that is built from a set of plug-in modules, and it provides a Java-based API that allows users to extend Protégé and build knowledge-based tools and applications, which implies that an ontology can be programmatically accessed. For the reason of compatibility and extensibility, Protégé is chosen as the main knowledge modelling tool for this present work.

4.2.3 Web Ontology Language (OWL)

OWL is a formal specification to define and instantiate ontologies, and has three sublanguages: OWL Lite, OWL DL (description logic) and OWL Full (Smith et al. 2004). OWL Lite is a simplified version of OWL that supports classification hierarchy but with limited expressiveness, for example only basic constraint features are available. On the contrary, OWL Full provides maximum expressiveness and much more freedom from its base (RDF), however without computational guarantees. In between, OWL DL is an intermediate version of OWL that maintains a balance between expressiveness and inference capability. OWL DL allows maximum expressiveness without losing computational completeness and decidability, in other words, all axioms are guaranteed to be computed and the reasoning process will finish in finite time. As a result, OWL DL is adopted in this research work.

OWL DL consists of three fundamental components: classes (concepts), properties (relations), and individuals (instances of the concepts) (Smith et al. 2004; Drummond et al. 2009). These will be referred as OWL classes, OWL properties and OWL individuals from this point. OWL classes are used to formally describe concepts in ontology, e.g. plastic is a concept of material. OWL properties establish “subject-predicate-object” relationships through two sub-types: the OWL object properties and OWL data properties. An OWL object property links two OWL individuals belonging to two OWL classes, while an OWL data property links an individual and data values. An OWL individual describes an instance in an interested domain (class).

As illustrated in Figure 16, “Pen_2011” can be an individual of class “Product”, “hasMaterial” can be an object property that links two classes “Product” and “Material”, and is instantiated as a linkage between two OWL individuals “Pen_2011” and “ABS_001”. In OWL, it is denoted as “hasMaterial (Pen_2011, ABS_001)”. An OWL data property,

"hasValue", instantiates a relationship between "Pen_2011_Weight" and a real number "20.33", i.e. "hasValue (Pen_2011_Weight, 20.33)". In either type of OWL property, the left hand is the domain and the right hand is the range so that to constraint the properties, e.g. OWL class "Product" is the domain of OWL property "hasMaterial", while "Material" is the range.

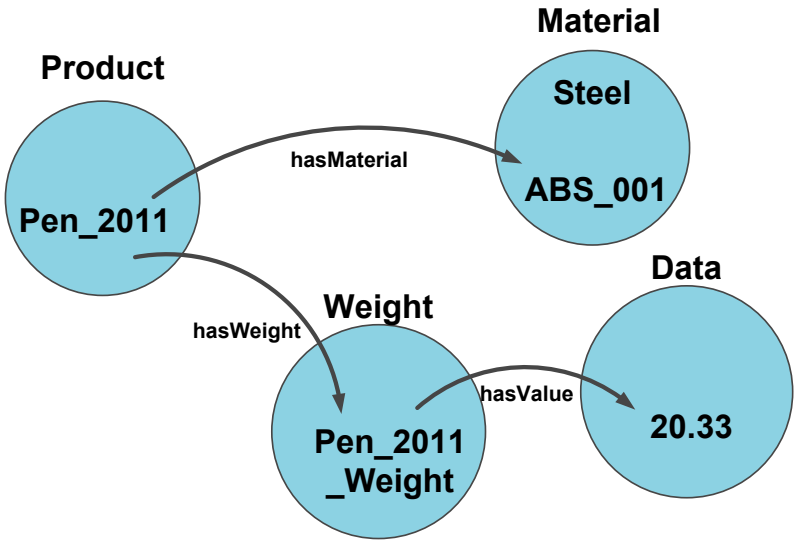


Figure 16 Illustration of OWL Fundamental Components: Class, Property and Individual

Based on these two primary property types, OWL uses property characteristics to enhance reasoning capability, as shown in Table 11. Furthermore, OWL also provides a mechanism called 'property restrictions' to further constrain the range of properties, in order to enhance the expressiveness and inference ability. Some OWL property restrictions are listed in Table 12.

Table 11 OWL Property Characteristics (Smith et al. 2004)

Property Characteristics	Description
Transitive property	If a property P is specified as transitive then for any x, y, and z: P(x,y) and P(y,z) implies P(x,z)
Symmetric property	If a property P is tagged as symmetric then for any x and y: P(x,y) if and only if P(y,x)
Functional property	If a property P is tagged as functional then for all x, y, and z: P(x,y) and P(x,z) implies y = z
Inverse property	If a property P1 is tagged as the owl:inverseOf P2, then for all x and y: P1(x,y) if and only if P2(y,x)
Inverse functional property	If a property P is tagged as InverseFunctional then for all x, y and z: P(y,x) and P(z,x) implies y = z

Table 12 OWL Property Restrictions (Smith et al. 2004)

Type of Restriction	Property Restriction	Description
Quantifier restrictions	Universal restriction (allvaluesfrom)	"Only" the specified class or data type allowed as the range, e.g. "hasMaterial (a, b) only Plastic".
	Existential restriction (someValuesFrom)	At least one such specified range, e.g. "hasMaterial (a, b) some Plastic".
Cardinality restrictions	Minimum cardinality	The keyword is "min", e.g. hasMaterial (a, b) min 3 Material.
	Maximum cardinality	The keyword is "max", e.g. hasMaterial (a, b) max 6 Material.
	Exact cardinality	The keyword is "exact", e.g. hasValue (a, b) exact 1 float.
"hasValue" restriction	"hasValue" restriction	The keyword is "value", for example: A author class "Chris, Linda, Chun" "hasAuthor (a, b) value Chun" implies that at least one author – Chun.

OWL also provides complex classes constructed with some operators, such as set operators (i.e. intersection, union and complement), or to make comparison between classes. Together with the previously introduced mechanisms, OWL again has these facilities to assist with the expressiveness and inference ability. However, OWL itself still has weaknesses, such as the absence of mathematical expressions and complex logical property chains. These issues are tried to be addressed by its complementary rule language and query languages, as described in the forthcoming sections.

4.2.4 Semantic Web Rule Language (SWRL)

SWRL (Horrocks et al. 2004; Stanford Medical Informatics 2011) is an expressive sublanguage of the rule markup language (RML) based on OWL sublanguages, OWL DL and OWL Lite. It offers a high-level abstract syntax to extend the OWL syntax. Therefore, SWRL allows users to write rules that complement OWL itself to provide more powerful deductive reasoning capabilities. Semantically, SWRL is much more powerful than other two query methods, such as regular expression (keyword search) and DL query (Protégé feature). Fundamentally, it is very similar to DL query in its basic rules. But it is more advanced in the means of mathematic operations with its built-in functions. A combination of queries can be processed from left to right. Results from computation at each step will be carried over for downstream processes. And it can also help to modify an ontology by computing new value to individuals.

SWRL Syntax

A SWRL rule consists of two parts, the antecedent part (body) and the consequent part (head). Each of them can be represented with at least one atom. The SWRL rule implies that if the antecedent is true, then the consequent must be true. An example rule in Figure 17 can be translated as if a geometric model "g" is defined with a type of material "Iron" and a manufacturer "doer" who has the facility for "SandCasting", and then it must be true that the "doer" is capable to manufacture geometric model "g". In this example, "hasMaterial (?g, Iron)" is an atom expression that consists of predicate symbol "hasMaterial" and followed with a number of arguments ("g" and "Iron" in this case).

```
GeometryModel(?g) ^ hasMaterial(?g, Iron) ^
    hasManufacturingFacility(?doer, SandCasting)
    -> isCapable(?doer, g)
```

Figure 17 Example of SWRL rule

For better expressiveness, SWRL has seven types of atom: 1) the class atom that uses OWL named classes as arguments; 2) the individual property atom that uses OWL individuals as arguments for an OWL object property; 3) the data valued property atom that uses one OWL individual and one data value as arguments for an OWL data property; 4) the different individual atom that differentiates two OWL individuals; 5) the same individual atom that defines two identical OWL individuals; 6) the data range atom that expresses a data range from specific datatype or a set of available values; 7) built-in atoms that express user-defined predication clause, in which if an antecedent composed of at least one argument is true, then the consequent is true. The antecedent can be defined as common mathematical operation, comparison operations or string operations. An example of a built-in atom shown in Figure 18 implies that if a part is manufactured using sand casting and its weight is less than 25 gram it is not feasible for this particular manufacturing process.

```
Part(?p) ^ hasManufacturingProcess(?p, SandCasting)
    ^ hasWeight(?p, ?weight) ^ swrlb:lessThan(?weight, 25)
    → Inapplicable(?p)
```

Figure 18 Example of comparison built-in atom

SWRL built-in atoms support OWL datatypes and customized datatypes, which makes a wide range of built-in libraries possible, and can also be extended by built-in implementers. Therefore, issues such as unit conversion and taxonomy search may be solved.

SWRL features

SWRL has many features that conform to OWL and further complement the expressiveness for logic. Some important features include:

- Open World Assumption: For example, if a PartWithHoles is defined as “Part (?p) ^ hasHole(?p, ?h) -> PartWithHoles(?p)”, it is certain that this part has feature of holes, but does not exclude the possibility of having slots too.
- Monotonic Inference: Same as OWL, SWRL supports monotonic inference only, namely, new instances can be created but existing entities can not be modified by SWRL rules.
- Decidability: SWRL improves the expressiveness of OWL, however at the cost of decidability. When reasoning over SWRL rules, the result may not decidable as opposed to OWL DL which guarantees termination of inference. Therefore, SWRL should be used wisely and only when the additional expressiveness is necessary.
- Debugging: SWRL rules can be potentially very complex, which make the debugging important to ensure that rules are correctly defined. This can be assisted with a SWRL-based query language Semantic query-enhanced web rule language (SQWRL).

4.2.5 Semantic Query-Enhanced Web Rule Language (SQWRL)

There is a well-accepted RDF query language SPARQL (Prud'hommeaux and Seaborne 2008) can be also used to query OWL due to the fact that OWL is built upon RDF. However, there is naturally incapability to serve OWL as it is natively designed for querying RDF (O'Connor and Das 2009). There also exists other query languages designed for OWL ontologies, such as OWL-QL (Fikes et al. 2004) and DIG’s ASK protocol (Bechhofer et al. 2003). Unfortunately, none of them has been maturely implemented to serve OWL as general purpose. SQWRL is therefore designed by Stanford Centre for Biomedical Informatics Research for making direct queries to OWL with a user-friendly syntax.

SQWRL (O'Connor and Das 2009; Stanford Medical Informatics 2010) (pronounced squirrel) is an native query language to OWL and is based on the SWRL. SQWRL takes the antecedent part of a standard SWRL rule as its pattern specification for a query and

the consequent part is replaced by a SQL-like[7] retrieval specification as illustrated in Figure 19. Therefore, no extra learning curve is required for SQWRL, which can be readily used to debug SWRL rules without any potential violation of OWL ontologies during development.

SWRL	*Part(?p) ^ hasManufacturingProcess(?p, MachiningProcess)* *→ MachinedPart(?p)*
SQWRL	*Part(?p) ^ hasManufacturingProcess(?p, MachiningProcess)* *→ sqwrl:select(?p, MachiningProcess)*

Figure 19 Example of debugging SWRL with SQWRL

SQWRL has two main features for improving expressiveness and semantics. One is a core operator "sqwrl: select" and some built-in operators (e.g. "sqwrl: count", "sqwrl: min", "sqwrl: max" and "sqwrl: sum") that provide a basic querying functionality to retrieve information. Also another feature is set operators that support closure operations to complete the weakness of open world assumption by explicitly define the operation scope. The collections include basic collection, grouping, aggregation, comparison and so on.

4.2.6 Ontology Modelling Methodologies

In order to assist in establishing ontologies by following a formal process, a growing interest in developing ontology modelling methodologies has been shown since the 1990s (Gomez-Perez et al. 2004). A general process in ontology modelling can be identified in four stages: indentify purpose, building the ontology (i.e. ontology capture, ontology coding, and integrating existing ontologies), evaluation and documentation. In the early stages, identifying the main concepts in an ontology is very critical to the merit of the knowledge base being developed. Uschold and King (1995) suggested ontology modelling methodologies can be classified into three strategies in terms of identifying the main concepts: top-down approaches, middle-out approaches, and bottom-up approaches.

Top-Down Approaches

In top-down approaches, the most abstract concepts are identified first then, specialized into more specific concepts. One top-down approach is SENSUS introduced by Swartout

[7] SQL (Melton and Simon 1992) SQL is a relational database query language, and stands for Structured Query Language.

et al. (1997). SENSUS is an ontology that aggregates more than 50,000 concepts based on dictionaries of multiple natural languages. To define a domain specific ontology by reusing a concept pool, “seed” words are used to walk through the existing ontology in reverse in order to reach the root of SENSUS. The "seed" terms provided by domain experts are the most general terms to identify entire sub-trees of the model that are relevant to include. All terms and the relations among them are collected on the route in order to establish the scope of the concerned domain, which provides a fast domain specific ontology modelling approach. As its nature, the prerequisite of this top-down approach is an enormous pool of concepts, and the approach is not suitable for modelling ontologies from scratch or adding a modular extension.

Middle-Out Approaches

In middle-out approaches, the most important concepts are identified first and then generalized and specialized into other concepts. One typical middle-out approach by Uschold and King (1995) addresses the basic concepts before any super- and sub-concepts are confirmed, in four stages (identify purpose, building the ontology, evaluation and documentation). In particular, the second stage involves knowledge capturing, knowledge encoding and knowledge integrating as illustrated in Figure 20. In each phase, many methods can be used, such as the Delphi method (Skulmoski et al. 2007) to assist with decision making by processing knowledge from a group of experts in an interactive and recursive way.

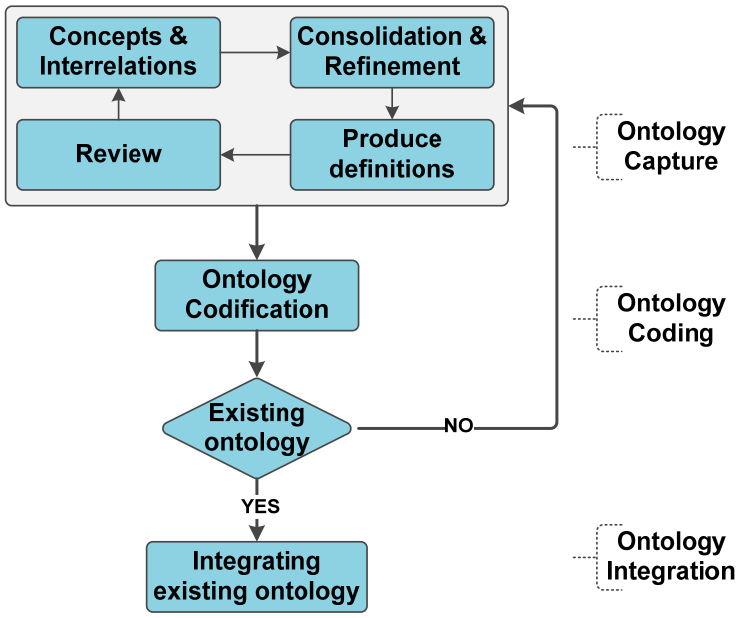


Figure 20 Three Processes to Build Ontologies by Uschold and King (1995)

Another middle-out methodology METHONTOLOGY (Fernandez-Lopez et al. 1997) is

suggested as the most mature ontology modelling methodology by Corcho et al. (2003). METHONTOLOGY has some parallels to Uschold and King's four-stage approach but is more detailed, and consists of a series of processes as illustrated in Figure 21. During specification, knowledge acquisition and conceptualization, many methods can be utilized, such as literature review, brainstorming, questionnaires (e.g. Delphi method) and so on, in order to initially consolidate the concepts (or classes) and interrelations (or properties). The integration process ensures the reuse of suitable existing ontologies to satisfy new requirements. The implementation is to formally codify these classes and properties as the results from previous processes. Then the ontology is evaluated, including to verify the completeness, consistency and redundancy and to validate the correctness of the logic. During the entire lifecycle, documentation is formally carried out in all processes to assist with the codification and record development for future references.

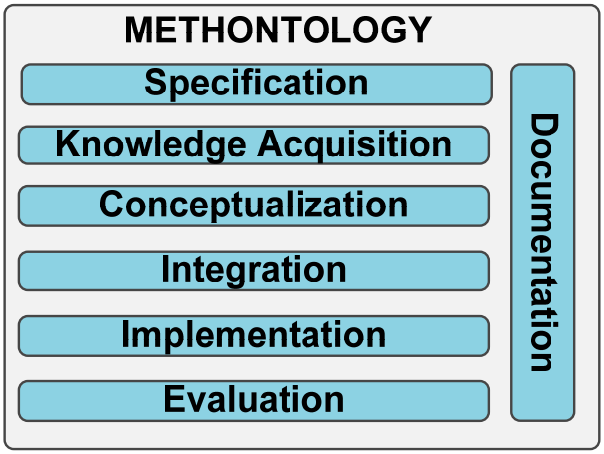


Figure 21 Processes of METHONTOLOGY (Fernandez-Lopez et al. 1997)

Bottom-Up Approaches

In bottom-up approaches, the most specific concepts are identified first and then generalized into more abstract concepts. Bernaras et al. (1996) introduced a bottom-up strategy KACTUS as shown in Figure 22. KACTUS is application oriented. It starts with defining the specification for a specific application and then builds a first ontology. It then tries to adapt this ontology in a more general way so that it can also serve another application. Iteratively, a consensual knowledge base for a collection of applications can be established and gradually refined (Gomez-Perez et al. 2004).

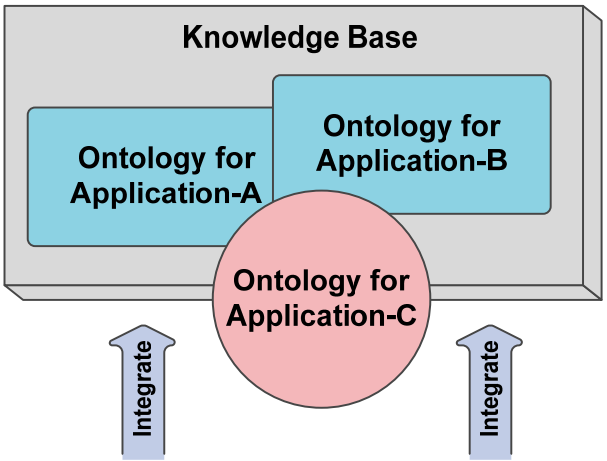


Figure 22 KACTUS Methodology (Gomez-Perez et al. 2004)

4.2.7 The State of the Art in Ontologies

In this section, the current status of ontologies is described, including their roles, applications in various areas, and how ontologies are used. Then, with the focus moved to mechanical engineering, how ontologies can potentially address the engineering design issues is explored.

Knowledge Acquisition

The natural role of ontology is knowledge representation, which is one of the processes related to a knowledge-based system. Knowledge acquisition is another important process which is the prerequisite to knowledge presentation (Gasevic et al. 2006) so that knowledge can be gathered, structured, organized and processed for a topic, a domain or a problem area of interest. In the author's viewpoint, there are two interfaces to the ontological knowledge: the *outer interface* for the knowledge base to capture knowledge from the external environment and the *inner interface* for users (both computer programs and human users) to access knowledge.

The outer interface handles the associativity between knowledge and the object being described. As mentioned in the previous sections, annotation as a natural information media has been widely used in web technologies, such as the annotation tool Annotea (Kahan and Koivunen 2001) for users to markup web pages, or the semantic annotation tool Magpie (Dzbor et al. 2004) that automatically parses the semantics of web pages. Magpie is then superseded by MnM (Vargas-Vera et al. 2002) to allow user interactions, so that knowledge can be captured in either an automatic or semi-automatic way. Shi and Setchi (2010) also proposed an automatic NLP approach to capture semantics from text documents. This approach was then improved and extended to automatically process

semantic information around images in order to aid engineering concept design (Setchi et al. 2011). Many more annotation approaches for manual or automatic information collection are reviewed by Li et al. (2009a; 2009b). As these articles suggested, annotations can be considered as a pragmatic mechanism to capture data/information from users or an external environment in the Web both manually or automatically, however they are often weak in handling knowledge.

With the focus on CAD, this refers to the outer interface that users can manipulate the design model with. There are many annotation approaches based on CAD systems. In the conceptual understanding and prototyping (CUP) approach (Anthony et al. 2001), semantics related to the functions, behaviors and structures of the parts in an assembly can be collected through a CAD system interface, however, it does not handle further detail of the CAD models, e.g. faces. Hisarciklilar and Boujut (2007) also explored annotation approaches based on 3D CAD models to aid engineering design, but the semantics is recorded as notes and comments in natural language rather than to a rigid formal specification and the authors have not suggested how they can be programmatically processed. Therefore, there is a need in CAD interfaces to external environments to enable rigid semantics to be recorded.

On the other hand, the inner interface handles the access to ontology models, for example ontology tools that users can use to directly access knowledge models or a computer agent that belongs to CAD system and can access the ontology models. In the first case, ontologies can be easily accessed through ontology modeling tools that have been introduced earlier, such as the Protégé ontology editor. However, the interface between ontologies and CAD systems for the second case is weak. And in order to smoothly convey ontological knowledge (not merely data/information) through the route from external environment, through CAD systems to the knowledge base, there is a need for integration of these three environments.

Knowledge Representation

Once knowledge is captured, it needs to be coded and stored while maintaining its reference to the CAD models. This implies the associativity[8] between non-geometry information and geometries. The ISO 16792:2006 standard (ISO 2006) established based on American Society of Mechanical Engineers (ASME) standard ASME Y14.41 – 2003

[8] Associativity refers to the established relationship between digital elements. (ISO 2006)

(ASME 2003) is a standard for product definition data, or product and manufacturing information (PMI), which allows the collection of all information in order to define and manufacture a product. This standard particularly addresses associativity by defining annotation practices for CAD systems, where practical requirements and constraints for annotation are specified both in general and in specific detail. The annotation in this particular case refers to dimension(s), tolerance(s), note(s), text or symbol(s) visible without any manual or external manipulation in this case (ISO 2006).

However, ASME Y14.41 is not designed for geometric representation. This gap is filled by the STEP standard, in which product data can be formally specified in its EXPRESS language and try to cover much broader coverage including geometric representation supported by most leading CAD systems (SCRA 2006). A database approach by El-Mehalawi and Allen Miller (2003a; 2003b) decomposes 3D CAD models expressed in STEP into elements and stored as database entries. This gives the ability for geometry elements to be associated with extra information at all levels of granularity. However the semantics can not be expressed among geometry elements. In general, both standards (i.e. the ASME Y14.41 and STEP) address semantics very weakly or not at all, and consequently they are incapable of reasoning actions.

Some research effort in the literature tries to provide semantic capability to these industrial standards. For examples, OntoSTEP (Krima et al. 2009) programmatically transforms STEP specified in EXPRESS into OWL-DL ontologies to enrich semantics. IfcOWL (Beetz et al. 2005; Beetz et al. 2009) develops OWL ontologies from EXPRESS in the building and construction field. Again, another work tries to translate the ISO 15926-2 standard coded in EXPRESS into OWL for integration of life-cycle data for process plants in oil and gas industry (a generic 4D model for PLC information) (Batres et al. 2007).

The aforementioned research addressed some particular issue for their specific application. However, none of them offers a comprehensive solution to manage and use knowledge base in an integral way. Some works focus on addressing associativity only, without considering downstream processes. Some focus on knowledge model transformation for semantic geometric representation, but without further exploration on KB integration and extension. Some emphasize on knowledge representation but not knowledge acquisition. In addition, it is weak in addressing MEV and how such knowledge can be reused and further processed, e.g. query and reasoning. In comparison with the Semantic Web or others, this is relatively underdeveloped in the mechanical engineering domain. Some research effort tries to stress this issue and will be described in the next section.

Semantic Interoperability and System Integration

As ontology is used to specify a conceptualization, it provides an explicit description of *taxonomy* (hierarchical concepts within a domain). The taxonomy classifies information entities into categories based on their properties, and together with the vocabulary establishes a conceptual framework for computers to process information, such as semantic information analysis and retrieval. This enables engineering information to be dynamically categorized and managed. One exemplar application of this feature is a semantic file system (SFS) (e.g. semantic instead of location (SIL) (Eck and Schaefer 2011), or layered ontology file system (Hung Ba et al. 2007)), or library system based on a faceted classification of tags (Giess et al. 2008; Wild et al. 2009). Again, Lim et al. (2009) introduce a semantic annotation approach to assist with the information searching and retrieval with the help of product family ontology, in which product variants are clustered based on product features.

In practice, ontology provides a *vocabulary* that defines the terms in a specific domain and provides logical statements to describe the relationships among them. This enables a computer to process a topic explicitly according to the context (e.g. taxonomy) in order to remove ambiguity (McGuinness and Dieter 2003). Therefore, a well-defined and well-structured ontology can be used for consistency checking in an application, i.e. to validate the information entity (e.g. check whether the property type is legal, or its value complies with restrictions). This also enhances the interoperability between different applications through its rigid semantic specification (Ciocoiu et al. 2001; Gašević et al. 2006b), e.g. taxonomy, vocabulary and inference rules. For example, an application A has an ontology that understands material type “iron”, and defines “cast_iron” as a type of “iron” having the property “usedFor” a manufacturing process “casting”, and another application B that does not understand “cast_iron”. But the application B is still capable to learn that “cast_iron” is a type of “iron” “usedFor” “casting”.

Taxonomies and vocabularies have been intensively used in the Semantic Web (Gašević et al. 2006b). Analogous to the Semantic Web, we believe that semantic techniques can also be applied to mechanical engineering, especially in the CAE field and their applications including CAD systems in order to aid semantic interoperability and integration. According to Patil et al (2005), semantic interoperability refers to automating data exchange according to its associated meaning among information resources throughout the PLC. For example, geometries and other recorded information relating to product design can be computable and understandable by users including both human and computer programs, therefore can be exchanged or for downstream processing.

Semantic interoperability has attracted intensive interest in recent years. The Product Semantic Representation Language (PSRL) (Patil et al. 2005) represents an ontology-based high level Interlingua that explicitly specifies terminologies in different applications, to support seamless semantic communication and system integration. There is also active research work on layered ontology architectures since the introduction by Jasper and Uschold (1999). A project by Zhu et al. (2009) is to aid data exchange and sharing between CAD/CAE applications through an ontology approach. The authors established three-layered ontologies to describe a common ontology, a domain specific ontology and an application specific ontology. In their case study, two CAD systems are modelled as application specific ontology, while one of them is populated with instances of geometric modelling features that are programmatically extracted from the CAD model itself. As a result, the knowledge in this geometric model processed in the format of that particular CAD system can be conveyed through the upper levels of ontologies and then understood by the other end (the other CAD system).

Unfortunately, the common weakness of the approaches described above is the absence of standards. As a consequence, it not only limits the richness and explicitness of semantics and the freedom of user behaviours, but also hinders interoperability among PLM applications (Rachuri et al. 2008). In order to improve interoperability, some approaches use industry standards as a basis to describe product definition data, terms and rules. Another layered ontology approach (Dartigues et al. 2007) establishes ontologies to describe modelling features of geometry complying with the STEP standard and therefore aids the integration of CAD and CAPP (Computer-aided Process Planning) by generating process plans based on the semantics extracted from CAD geometry information. Furthermore, Posada et al. (2005; 2006; 2006) try to improve virtual reality (VR) capabilities of CAD systems through semantically-rich and STEP-compliant geometric representations.

There are certainly many other research works that have been carried out in this field, such as describing product family or characteristics ontologically to help product modelling (Lim et al. 2009; Bock et al. 2010), supporting PLM application integration including CAD (Peachavanish et al. 2006), CAM (Shen et al. 2006), ontology mapping and standard-to-ontology transformation (Beetz et al. 2005; Egaña et al. 2008; Beetz et al. 2009; Krima et al. 2009), reasoning processes over semantic knowledge (Zhu et al. 2010), and the function-oriented ontology approach Funnotation (Kitamura et al. 2006).

4.2.8 Concluding remarks

Ontology related issues have been explored and discussed in this section, including the ontology definitions ontology specification language and tools. As a result of this review, one ontology language, OWL, is then particularly described in more detail, also its complementary rule language SWRL and the query language SQWRL. Ontology modelling tool Protégé shows its merit over the others in suiting this present work. The knowledge modelling methodologies are also reviewed and categorized into three conceptualisation strategies: top-down, middle-out and bottom up approaches to fit in different situations or purposes.

The development and applications of ontologies have also been reviewed, particularly in the engineering domain. Some research issues are identified mainly in three aspects: the interface to knowledge acquisition and process; the ontology architectures for knowledge management and the unsolved system integration issue. Furthermore, advanced reasoning mechanisms and the applications are still notably under-developed in engineering domain.

Regarding the question Q2 of the hypothesis H2 stated earlier in this chapter: how knowledge and information can interoperate, it can be concluded as following:

H2-Q1: How ontologies can be used to define semantics?

> The definition of semantics can be specified using classes (concepts), properties (relations) and individuals (instances). Many ontology languages are able to provide the formal descriptions.

H2-Q2: How ontologies can be used to manage semantics in respect of incorporating various ranges of expertise?

> Ontology concepts for domain expertise can be managed as modularized knowledge with architectures such as FSB structure, or layered ontology architecture. The ontology modelling methodology also addressed knowledge management with different strategies that suit diverse situations.

H2-Q3: How ontologies can be used to process the defined semantics?

> Ontology metadata can be semantically searched and retrieved. Ontologies provide logic descriptions so that it can be reasoned with. This includes consistency checking for both schema and logics.

H2-Q3: How ontological technologies have been used in the engineering field and what are the weaknesses of current applications?

> Ontologies have been applied in many fields, especially in the Semantic Web, however are less developed in mechanical engineering. This is implied by the fact that there is no successful ontology-driven knowledge base supporting CAD systems. On one hand, most approaches support ontological descriptions at a part level, but incapable of addressing all levels of granularities for CAD models. On the other hand, most approaches provide an inner interface for users to directly manipulate ontology metadata, but not the outer interface to allow users to establish associations between CAD models and ontological knowledge. Further more, it has been claimed by some researchers that data models in STEP can be transferred into OWL ontologies. This enables that CAD models can be redrawn in OWL, and consequently opens the possibility of consistent semantic anchoring.

According to this discussion, the support of ontologies in engineering KIM can be depicted in Figure 23, in which issues for targeted media types (e.g. text, multimedia and 3D models) and degree of formality in knowledge representation can be fully supported. However other issues such as using ontology to provide broader coverage in granularity, robustness of association between information entities, knowledge access (e.g. storage of knowledge data, query and retrieval), downstream knowledge process, and knowledge modelling methodology remain. It is only affirmed that ontologies have positive influence, but gaps still appear in order to establish a general purpose engineering knowledge management environment to assist CAD systems.

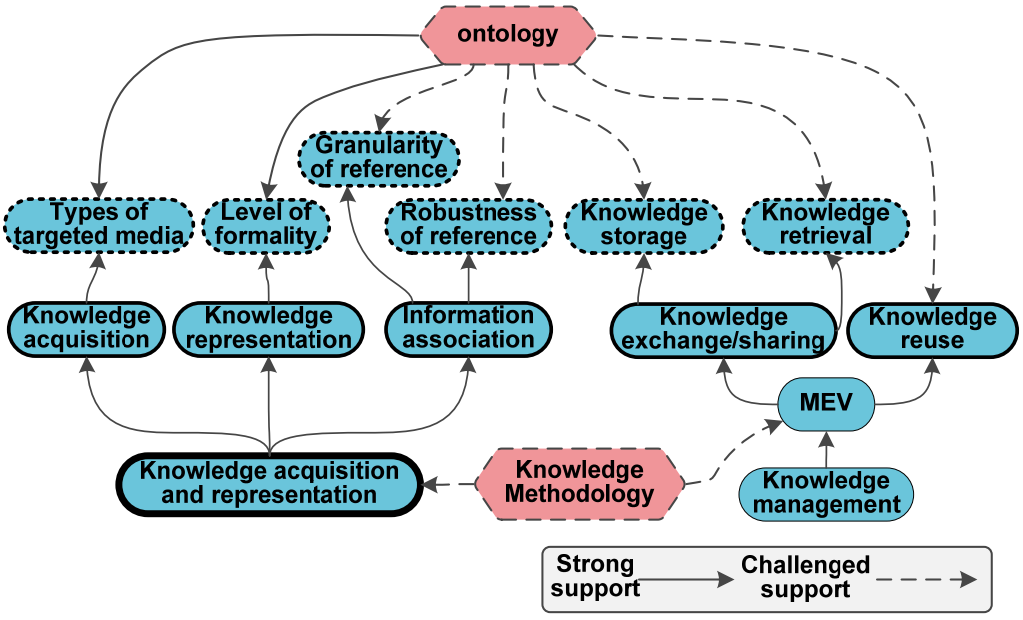


Figure 23 Impact Model for the Support of Ontologies and Knowledge Modelling Methodologies

4.3 DRM Models and Concluding remarks

This chapter reviewed the literature in annotation and ontology technologies and their applications with the focus on the mechanical engineering domain. It has been found that annotation and ontology technologies have valuable features and potential for knowledge acquisition, representation, management and downstream processing. However, both of them have their own weakness in achieving the main aim and objectives of this present work alone but show potential to complement each other. Referring back to the research question – Q3 how engineering services/tools can be integrated with CAD system, based on its associated hypothesis H3, it is concluded as follows:

H3-Q1: How annotation and ontology can aid CAD system respectively?

Annotation can play the role of media that captures extra data/information from CAD models or users either automatically or manually. Annotations can be in any degree of formality, and stored stand-off so as to make the repository portable and independent from the CAD system. Ontology can be used to represent both static and dynamic knowledge, and to support semantic knowledge processing.

H3-Q2: Whether annotation and ontology complement each other and how?

As discussed in H3-Q1, annotation may construct data but is incapable of constructing knowledge, while ontology is theoretically good at knowledge representation but lacks the ability to interact with CAD systems. This implies a chance for annotation, ontology and CAD system to co-operate as depicted in Figure 24, in which the interface, knowledge and data presentation fills the gaps raised in previous reference and impact models.

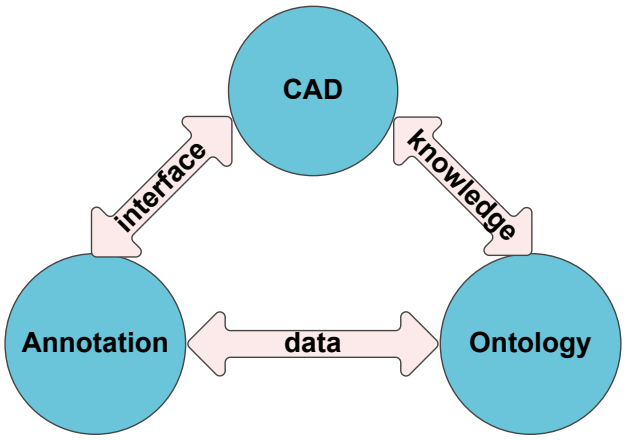


Figure 24 Possible Synergy among CAD, Annotation and Ontology

A Complete Reference and Initial Impact Model and Criteria

According to the literature and the preliminary experimental work, the potential impact of annotation (Figure 15) and ontology (Figure 23) can be merged into the previous reference and impact model (Figure 9). Thus the reference and impact model is updated as Figure 25. To keep the diagram concise, reference sources are omitted as they have been mentioned before. As a result, a set of Success Criteria (SC) and Measurable Success Criteria (MSC) for a desired support can be produced as listed in Table 13.

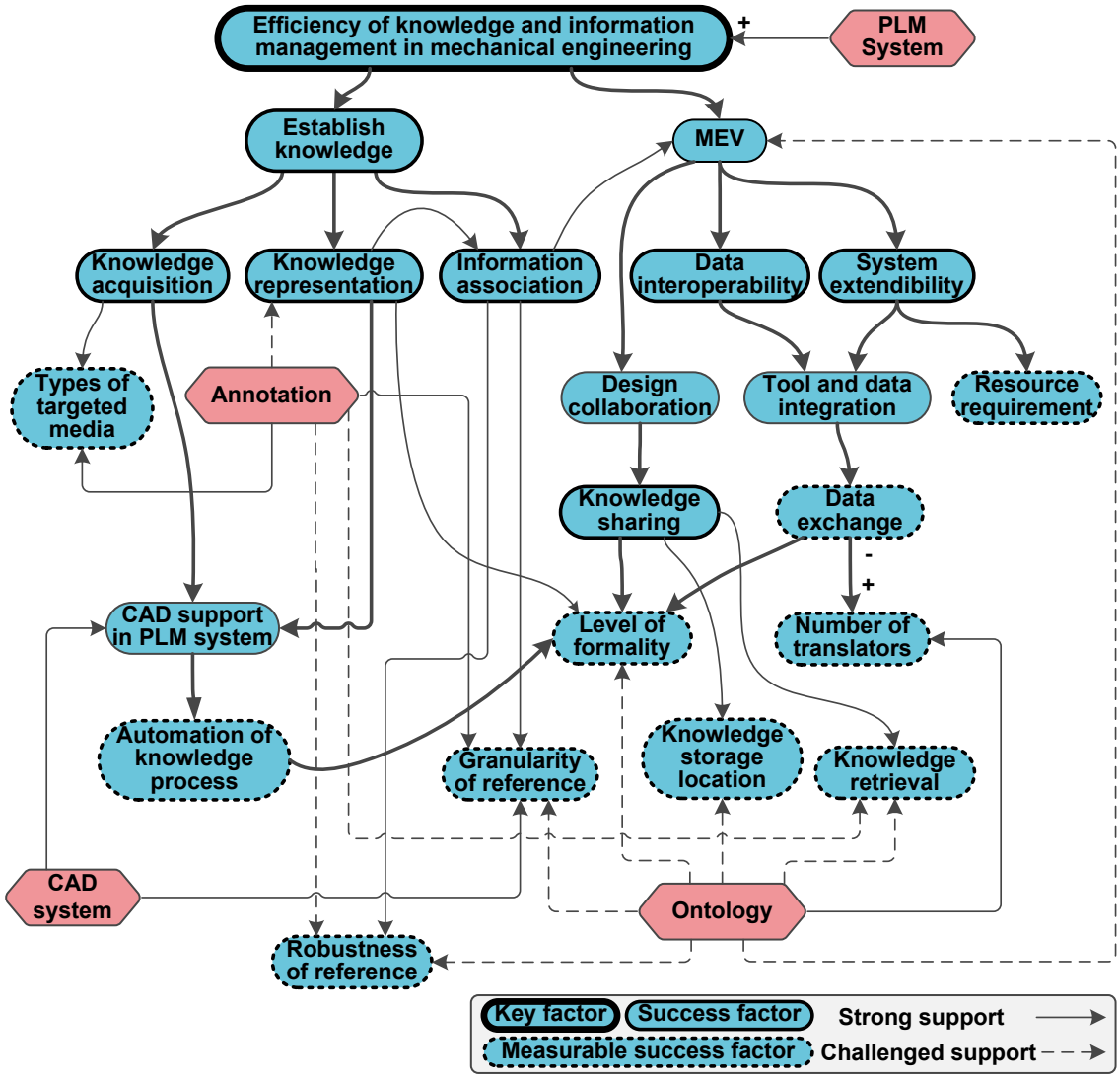


Figure 25 Updated Reference and Impact Model with Potential Support of CAD System, Annotation and Ontologies

Table 13 Success Criteria and Measurable Success Criteria

Success Criteria		Measurable Success Criteria	
SC1	Knowledge can be acquired.	MSC 1	Knowledge can be captured through CAD system.
SC2	Knowledge can be stored.	MSC 2	Knowledge can be formally specified and saved in a knowledge base.
SC3	Knowledge can be represented.	MSC 3	Knowledge can be represented through CAD system.
SC4	Knowledge can be associated with design model.	MSC 4	Knowledge can be associated with CAD model in different granularities.
SC5	Knowledge can be shared.	MSC 5a	Knowledge can be retrieved from knowledge base.
		MSC 5b	Knowledge can be shared within the total system.
SC6	Data can interoperate.	MSC 6	Knowledge data can be exchanged with external systems.
SC7	System can be extended.	MSC 7	Tools can be integrated with few resources required.
SC8	Knowledge process can be automated.	MSC 8	Automation of knowledge process can be achieved.

The Implications for Support Development and Evaluation

It is the author's viewpoint that engineering knowledge can be used more intensively and effectively to aid engineering design and analysis by adequately addressing these issues in an integrated way. It can be a systematic, standardized and evolvable framework to represent, manage and exchange knowledge with robust associations among information entities so that engineering knowledge can collaborate among MEV and across computational environments. And this framework can be evaluated whether the current situation is improved against the MSC. Based on the reviewed literature and experimental work so far, a general purpose semantic annotation approach to aid CAD systems is proposed and will be introduced in the forthcoming chapters.

Chapter 5 OntoCAD

In the previous chapters, it has been found that there are research gaps in current CAD systems. These include knowledge representation with robust associativity, knowledge management for incorporating MEV, semantic interoperability and system extendibility. It is argued that annotations, ontologies and international standards hold the promise to aid such gaps, but need to be further improved in an integrated systematic way in order to develop their potential.

In order to address the aforementioned challenges, an **Onto**logy-driven semantic annotation framework for **CAD** systems (**OntoCAD**) is proposed as a general framework to assist with engineering processes by incorporating MEV. In this framework, knowledge is captured, represented and managed in a modularized knowledge base, on which a platform-independent computational environment can be established. The modularization ensures the system extendibility as EVs are collectively integrated as a whole and evolve over time. With more and more new EVs and engineering applications developed, also with legacy applications integrated, the services have the potential to cover broader aspects throughout the entire PLC. The knowledge explicitly conveyed among MEV can improve interoperability and support collaboration among varieties of engineering teams. As the knowledge base is built using the formal ontologies, some reasoning processes can be automated to derive new knowledge or lead to downstream processes. In a nutshell, it is a formal semantic annotation system that assists users to ‘speak’, to ‘discuss’, to ‘remember’ and to ‘think’ within the PLC.

5.1 Overview of OntoCAD

To achieve this goal, the OntoCAD system architecture is proposed referring to the DL system architecture by Horrocks (2005), which consists of three layers: knowledge base, inference system and interface (Figure 26). To some extent, this is an independent KBS architecture. In this present work, this architecture is adapted to embed the knowledge base to the CAD system as illustrated in Figure 27.

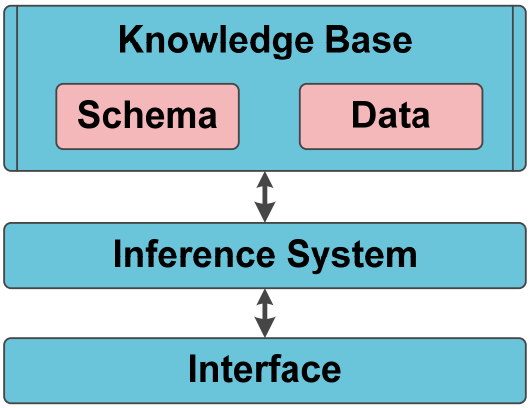


Figure 26 DL System Architecture by Horrocks (2005)

The OntoCAD system architecture is composed of three main modules: the OntoCAD Graphical User Interface (OGUI), the OntoCAD Knowledge Base (OKB) and the OntoCAD MEV Agent (OMA). In contrast to the approach by Horrocks, the CAD system interface is treated differently from other engineering application interfaces. This is because the CAD models are primary resources while all other non-geometric knowledge is secondary resources referencing the primary ones. Another improvement is the OMA that is a combination of the inference system and interface. As a result, the OMA can better handle semantics and interact with other modules. Furthermore, parts of the interface function are re-assigned to the knowledge base – the OKB. Therefore, the system has three interfaces to interact with CAD systems (upstream), the knowledge base (horizontal), and users or applications/services (downstream). Each module will be briefly introduced in following subsections.

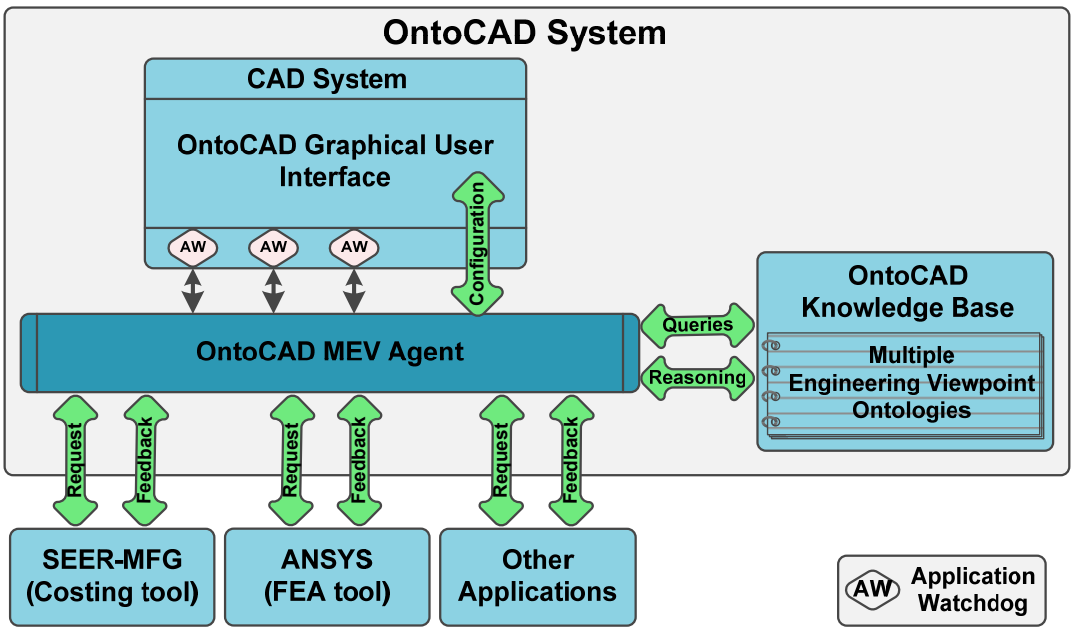


Figure 27 Overview of OntoCAD System

5.1.1 OntoCAD Graphical User Interface (OGUI)

The OGUI module is an outer interface. It tries to address issues for the knowledge acquisition and annotation anchors. The features include establishing associativity for primary resources, a dynamic annotation graphical user interface (GUI), support of a dynamic rendering system, interaction with both human and computer audience, and a monitoring agent – application watchdog (AW).

This GUI is embedded into the CAD system, so that end users can interactively annotate MEV knowledge (see Section 5.1.2) to the primary resources – CAD models. In other words, this is to set the annotation anchors. Theoretically, anchors can cover any level of granularity as a feature inherited from STEP, i.e. the anchor can range from fine grain (e.g. a point or a face) to coarse grain (e.g. a part and an assembly); it can also possibly be multi-point anchoring that references multiple geometry elements, or multi-directional anchoring that improves the traceability for annotation data.

The OGUI may dynamically change the GUI depending on the users' viewpoint. In other words, the user input interfaces may be different according to the users' context, and this is dynamically driven by the ontological knowledge base. For example, users are allowed to select kilograms when annotating weight, while they are guided to select 'square meter' when annotating area. This dynamic configuration of the GUI is in fact largely driven by the ontology module rather than laborious programming in the software, which will be described in the forthcoming section.

On the other hand, the annotation should be retrieved and rendered dynamically according to specific queries. Queries to the knowledge base may vary in each particular case, therefore the retrieved knowledge should be different too and not necessarily to retrieve all information related to the anchors being inquired. Instead, the result should conform to the query context and should be explicit.

Another important responsibility for the OGUI is to interact with the inner interface of the OMA (Section 5.1.3), which includes:

- Data exchange: send the captured knowledge to the knowledge base for storing, and receive the retrieval from the knowledge base.
- Perceive the interface configuration instructions (ICI) to guide the dynamic interface.

- The cooperation with the AW: the AW is the monitoring agent assigned by the OMA for a particular task, e.g. monitoring the status of a specific cost query.

5.1.2 OntoCAD Knowledge Base (OKB)

The OKB is the kernel of the OntoCAD system, which provides all AI knowledge foundations. It has some key features: stand-off annotation storage, capability to accommodate both structure and freestyle annotation and to hold engineering semantics.

Since stand-off annotation features have the advantage of easy maintenance and a high level of portability, this strategy is adopted in the present work. Knowledge is represented as ontologies including engineering concepts and annotation data as instances of ontologies. Therefore the knowledge base acts as a data repository. The ontologies will be STEP-compliant, especially for geometric representation, which contributes to robust associativity while not losing the freedom in controlling data. In the present work, a rigid formal data structure is adopted, but does not exclude the possibility to accommodate freestyle annotation data. This opens the possibility to add general text strings in natural language or to assign an identifier referencing external resources.

The most important feature is the knowledge management approach that incorporates MEV knowledge and the ability to support semantics. The KB has a three-layered MEV ontology architecture that will be described in Section 5.3. It is the driver to answer all the queries from the OMA, including data request and ICIs.

5.1.3 OntoCAD MEV Agent (OMA)

The OMA plays the role of intelligent broker that coordinates with all other modules: it interacts with the upstream CAD systems via the OGUI; it interacts with the kernel – the OKB - horizontally; and it interacts with downstream engineering applications.

In the first case, the OMA perceives the context of the end users according to their chosen EV. In this context and according to particular annotating actions or queries, the OMA will consult the OKB and give appropriate ICIs. It is also in charge of sending information or retrieving information to/from the CAD system, for example, to apply anchors to CAD models, or to extract data from CAD models, such as dimensions, volumes and materials. On the other hand, the OMA also controls AWs. This involves two steps: to define rules for an AW in a particular case by the end users; and to observe the status of the AW and report whether the query has been satisfied by the OMA.

In the horizontal direction, the OMA submits reasoning queries to the OKB according to requests from both CAD system and downstream engineering applications. Reasoning actions are classified into three main categories: factual reasoning, conceptual reasoning, and methodological reasoning. For the details please refer to Section 5.5.

In the downward direction, the OMA handles requests from all integrated engineering applications. In order to achieve this, it maintains a register of services, so that the OMA is self-aware, and the end users are also aware of all EVs and the integrated applications/services. Therefore, the query can be appropriately answered and knowledge can be conveyed through all ontologies.

Based on this overall system architecture, the rest of this chapter will describe in more detail on how annotation data are structured, how knowledge is organized, modelled and maintained, and how knowledge is used.

5.2 OntoCAD Annotation Data Structure

Since annotation consists of two components: the annotation anchor and annotation content, the annotation data refer to the data that specify these two components. This section will focus on the annotation data structure that provides a robust anchoring mechanism for knowledge acquisition, thus supporting knowledge representation, which establishes associativity between CAD model and additional engineering knowledge, and providing a stand-off annotation strategy to ease data maintenance.

5.2.1 OntoCAD Annotation Data Language Stack

Before diving into the detail of the annotation data, the language to “speak” annotation need to be decided. As requirements, the language should have the potential to be widely supported in PLM systems, portable and extendible. Considering the de facto web standards (e.g. XML and OWL etc), and their capability, inspired by the Semantic Web Stack (Berners-Lee 2000), an OntoCAD annotation data language stack is proposed. As depicted in Figure 28, towards the top left corner is the interface to the external environment, while towards bottom right corner is the foundation.

According to the nature of OWL, the bottom layers of this architecture are designed on top of XML and in turn on URI and the Unicode character set. As previously mentioned, XML is one of the most widely supported format in PLM systems (Cheung and Schaefer 2010), using XML as the foundation, OntoCAD can benefit from its popularity and the extendibility in the nature of XML language itself. This also makes the knowledge base

data exchangeable with the Web documents.

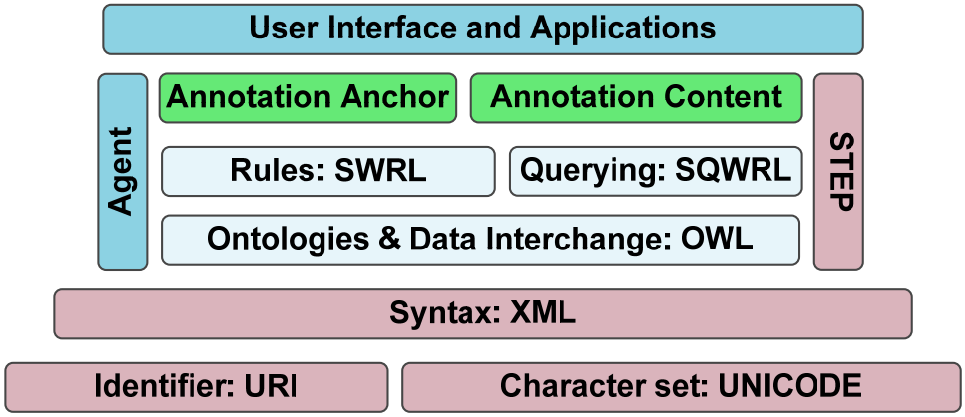


Figure 28 OntoCAD Annotation Data Language Architecture

The next higher level is the OWL ontology language that represents engineering knowledge, which is potentially exchangeable with other knowledge systems. As OWL is designed to specify ontologies, and also to instantiate instances (i.e. data in ontologies), OWL can be used as an annotation data specification language to describe both annotation components: the annotation anchor and annotation content. The fundamental annotation data structure will be next described in Section 5.2.2. As a result, annotation constituents are stacked on top of OWL and their rule language SWRL and querying language SQWRL. Therefore, the annotation data can be queried and reasoned under the control of the agent OMA with the assistance of SWRL and SQWRL. Coordinating through the OGUI and the OMA, data can be exchanged and knowledge can be conveyed and reused among CAD systems and integrated tools/services.

Furthermore, to establish the consensual knowledge among MEV and applications, STEP as a widely accepted standard is adopted in OntoCAD so that the proposed system can provide consensual terminologies to external and internal systems (e.g. geometries, data types, etc…), thus further improving the interoperability. A knowledge base therefore can be established upon and controlled by the agent and can enable downstream processes. For the standards that have been implemented into OntoCAD ontologies please refer to Chapter 6.

5.2.2 The Basic OntoCAD Annotation Data Structure

As mentioned earlier, the annotation data in this present work are specified in OWL DL. An OWL specified DL ontology can be conceptually divided into three components: the Terminology box (Tbox), the Relation box (Rbox) and the Assertion box (Abox) (Fokoue et al. 2006). The TBox statements describe a conceptualization – a set of concepts, such as

subsumption (e.g. Man subClassOf Person) and equivalence (e.g. Man equivalentClass MaleHuman). The Rbox statements describe relationships about roles and role hierarchies (e.g. hasSon subPropertyOf hasChild). The Abox contains role assertions between individuals (e.g. hasSon (Mary, John)) and membership assertions (John: Man). Tbox and Rbox together construct a set of *axioms* for the ontology skeleton, namely the structure of the knowledge model (i.e. schema), while Abox constructs a set of *facts* for some particular concrete situation, which fills instances into the skeleton (i.e. data). Based on this theory, the annotation data structure is proposed as illustrated in Figure 29.

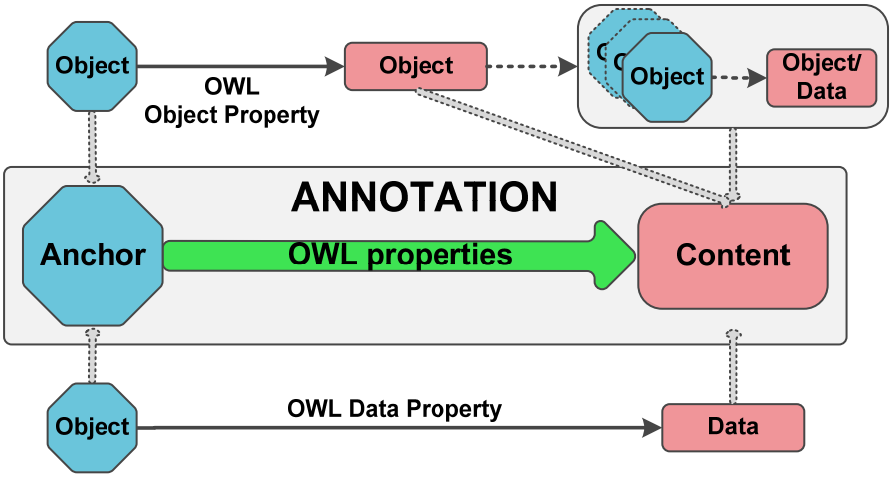


Figure 29 OntoCAD Annotation Structure

The annotation anchors are geometric elements instantiated as OWL individuals, the annotation contents can be filled with OWL individuals or data values, while the associativity between anchor and content is maintained by OWL properties. There are three types of annotations in terms of its structure: the data, object and annotation chains.

The Object Annotation

An *object annotation* is based on OWL object properties, uses an OWL individual as anchor (domain of an OWL property) and fills its content with OWL individual(s) (range of an OWL property). In the case of multiple anchors, the anchors can be a set of OWL individuals. The example annotation entry constructed by an OWL axiom (Figure 30) implies “BODY_1” has material “ABS”, in which “BODY_1” is an OWL individual of type “GeometryElement”, “hasMaterial” is an OWL object property, and “ABS” is an OWL individual of type “Material” defined elsewhere, and the types are predefined elsewhere in the ontology.

```
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#hasMaterial"/>
    <NamedIndividual IRI="#BODY_1"/>
    <NamedIndividual IRI="#ABS"/>
</ObjectPropertyAssertion>
```

Figure 30 Example of Object Annotation

The Data Annotation

A *data annotation* is based on OWL data properties, uses an OWL individual (or OWL individuals in the case of multiple anchors) as anchor(s) (domain of an OWL property), and the data annotation content is filled with data value(s) (range of an OWL property). The example annotation entry (Figure 31) implies "ABSRawMaterialCost" has value of "1.7019", in which "ABSRawMaterialCost" is an OWL individual of type "RawMaterialCost" defined elsewhere in the ontology as a subclass of "MaterialProperty" and "1.7019" is a value of datatype "double".

```
<DataPropertyAssertion>
    <DataProperty IRI="#hasRawMaterialCostValue"/>
    <NamedIndividual IRI="#ABSRawMaterialCost"/>
    <Literal datatypeIRI="&xsd;double">1.7019</Literal>
</DataPropertyAssertion>
```

Figure 31 Example of Data Annotation

The Annotation Chain

The third type *annotation chain* is a mixed properties type that is a combination of both OWL object properties and OWL data properties. As shown in the top section of Figure 29, it is chained properties that always start with an object annotation and linked further by a number of object annotations or one data annotation as the last node. To be noted, once a data annotation occurs in the chain, the annotation can not link further because a data value can not be associated with downstream objects. For the instance in Figure 32, "ABS" and "ABSRawMaterialCost" may have sub-links, but the value of "1.7019" cannot be linked any further.

```
BODY_1 → hasMaterial → ABS
    → hasRawMaterialCost → ABSRawMaterialCost
    → hasRawMaterialCostValue → 1.7019
```

Figure 32 Example of Data Annotation

Apart from the classification in terms of OWL property types, this annotation data structure

can also be classified as *direct annotation* or *indirect annotation* (Figure 33) in terms of the associativity to primary resources – the CAD models.

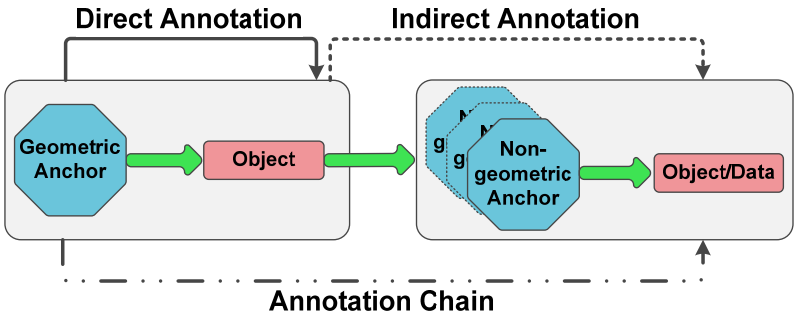


Figure 33 OntoCAD Direct and Indirect Annotation Types

Direct Annotation

Direct annotation refers to the annotation whose anchor is set to OWL individual(s) representing geometry element(s) (e.g. point, edge, face, and hole represented as OWL individuals).

Indirect Annotation

Indirect annotation refers to the annotation whose anchor is set to OWL individual(s) representing arbitrary object(s) other than geometry element(s), such as any node within the example in Figure 32 other than the initial node "*BODY_1* → *hasMaterial* → *ABS*".

5.2.3 Standard-compliant Annotation Anchoring Mechanism

STEP is an extensive family of product definition standards, covering many aspects, such as geometric representations, data modelling languages, data formats and so on in many industrial fields including automotive, construction and many others. The primary reason for adopting STEP is to establish consensual knowledge, especially geometric representation, therefore to provide a consistent semantic infrastructure to associate geometric model and engineer information. This present work does not try to achieve an automatic transformation from STEP to ontologies, nor does it try to cover the complete standards, as these are not the central work of the current research and have been tried by other researchers as described in Section 4.2.7, such as work by Krima et al. (2009) and El-Mehalawi and Allen Miller (2003a).

STEP standards are manually transformed into OWL ontologies, in other words, OWL ontologies are manually modelled to comply with STEP standards. Two aspects need to

be covered in a transformation: the specification language and the data (e.g. B-rep geometric representations and schema).

In the first aspect, the STEP description method defined in EXPRESS (ISO 1994b) corresponds to Tbox and Rbox (i.e. the classes and properties), rather than ABox (i.e. individuals) in OWL ontologies. According to the study by Krima et al. (2009), most basic concepts can be successfully mapped from ISO 10303-11:1994 – EXPRESS to OWL as shown in Table 14. Apart from those listed, OWL can provide richer semantics than EXPRESS by more complex restrictions constructed with OWL axioms and rules.

Table 14 Concept Mapping Between Languages EXPRESS and OWL

EXPRESS	OWL
Entity	Class
Subtype	Subclass
Attribute with an entity type	ObjectProperty
Attribute with a simple data type	DataProperty
Optional attribute	Universal restriction
Attribute with an aggregation type	Cardinality restriction

In the second aspect, the data for geometric representation and other data schema need to be considered. For ontological geometric representation STEP ISO 10303-203:1994 (ISO 1994f) is followed, and for data types ISO 10303-21:1994 (ISO 1994c) is followed. According to the classification of STEP conformance in an implementation, six classes are defined by the standard as described in Table 15.

Table 15 STEP Standard Conformance Classes (ISO 1994f)

Conformance Classes	Description
Class 1	Configuration-controlled design information without shape.
Class 2	Class 1 and shapes represented by geometrically bounded wireframe models, surface models, or both.
Class 3	Class 1 and shapes represented by wireframe models with topology.
Class 4	Class 1 and shapes represented by manifold surface models with topology.
Class 5	Class 1 and shapes represented by faceted B-rep.
Class 6	Class 1 and shapes represented by advanced B-rep.

Among these classes, Class 1 is the prerequisite needed to be conformed to by any class from Class 2 through Class 6. Since B-rep geometric representation dominates in CAD

systems and as a fact that Class 6 is applied in many leading CAD systems, such as NX and CATIA the highest conformance option Class 6 is chosen and applied in the experimental work (Chapter 6).

STEP entities can be classified into categories according to their functions (PDES Inc. 1998) as shown in Table 16. In each category, only the main entities are listed. For a complete list of entities that complies with conformance Class 6 of STEP ISO 10303-203:1994 please refer to Appendix 1.

Table 16 Function Categories of STEP Entities with Examples

Information Categories	Functions	Example Entities
General	AP Identification and Contexts	application_context application_protocol_definition
People and Organizations	People	person
	Organizations	organization
	Roles	person_and_organization
Dates and Times	Dates	date
	Time	local_time
	Roles	date_and_time, date_time_role
Approvals	Approval	approval_person_organization
Security	Security	security_classification classification_officer
Units of Measure	Units	length_unit, solid_angle_unit, area_unit, mass_unit, volume_unit
Shape	Units for Shape	geometric_representation_context
	Shape Aspects	shape_aspects
	Brep Models	advanced_brep_shape_representation
Parts in AP 203	Identifying Parts	Products, product_definition_formation
	Categorizing Parts	product_related_product_category
	Relating Specifications to Parts	cc_design_specification_reference
	Relating Parts to Contracts	cc_design_contract
	Renumbering Vendor Parts	supplied_part_relationship
	Alternate Parts	supplied_part_relationship
	Assemblies and Shape	shape_representation mapped_item
	And others	See (PDES Inc. 1998) and Appendix 1

Since STEP files can be imported by majority of leading CAD systems, STEP-compliant B-rep models are adopted as the semantic anchor representation in supporting CAD systems. For example, *advanced_face* is a term in STEP (ISO 10303-203:1994 (ISO 1994f)) to define the precise meaning of topologically bounded surface, for which attributes are illustrated in Table 17, where '*name*' is used as the *annotation anchor identifier* (AAI) assigned by OMA through OGUI, '*bounds*' describes its constituents and *face_geometry* defines the type of face, e.g. *elementary_surface*, or *swept_surface*. Each attribute may be defined in other classes.

Table 17 Attributes of STEP Term advanced_face

Attribute	Entity Type in OWL	Entity in STEP
name	entity name (STRING)	representation_item
bounds	SET OF face_bound (CLASS)	face
face_geometry	surface (CLASS)	face_surface

The proposed OntoCAD system tries to deal with three levels of granularity in the annotation anchors: G1 (body), G2 (face or faces) and G3 (edge or edges). Theoretically, this can be readily expanded to cover other levels of granularity, including higher levels – assembly, or lower levels – vertex, point and so on. However, it is currently focused on G1, G2 and G3, and the implementation will be described in Chapter 6. Since a geometric model specified in STEP can be redrawn as an OWL ontology, the geometric model and its constituents can be used as geometric anchors as illustrated in Figure 29, to which non-geometric engineering information can be attached through OWL properties. The three levels of granularity are listed in Table 18. With this support, geometric models can be annotated in detail from part, surfaces down to edges.

Table 18 Three Levels of OntoCAD Annotation Anchoring Granularity

Annotation Anchor Granularity	Annotation Anchor (OWL Class)	Annotation Anchor Identifier (OWL Individual)
G1	manifold_solid_brep	BODY_1
G2	advanced_face	FACE_23
G3	edge_curve	EDGE_103

5.2.4 Concluding Remarks

The immediate advantage of this data structure is that it eases knowledge maintenance since annotation data is stand-off from target CAD parts however maintaining robust association. Thus this structure enables the knowledge base as a whole to be entirely portable to any CAD systems that use STEP-compliant geometric representations.

The second advantage can be the general applicability. Since geometry and its constituents are specified as part of instantiated OWL ontologies, thus to be used as anchors, the same mechanism can be readily applied to anchor other levels of granularity, such as vertexes and assemblies, other types of product definition documents, such as text documents and multimedia documents. This can be easily achieved using URI as a data value of an annotation content pointing to external documents. Therefore, this anchoring mechanism is ideal to but not limited to CAD parts, which improves the evolvability of complex systems.

Another major advantage can be the ease of product data exchange due to the fact that annotation data is specified in a unified language. Adopting the OWL language as an annotation data specification language opens up this possibility. Furthermore, the fundamental schema conforms to a widely accepted industrial standard family – STEP, which further improves the interoperability and potential of tool integration ability with the consensual knowledge among participants.

Having described how knowledge can be captured, represented, associated and stored as ontologies, how to manage the represented knowledge will be introduced in the next section.

5.3 OntoCAD Knowledge Base

As previously introduced, the Tbox, Rbox and Abox in DL ontology construct a knowledge base. Apart from this schema and data representation, we also concentrate on how the captured annotation contents are represented as knowledge and managed in order to aid in engineering design and enable downstream processing. Engineering design is often described as a cyclic process in which physical concepts are proposed and developed, and their fitness for purpose is evaluated from MEV, such as performance, structural integrity, manufacturability, cost and so on (Sommerville and Sawyer 1997; KwangHoon et al. 2003; Davies and McMahon 2006; Pahl et al. 2007). Therefore, a primary goal to knowledge management is to aid collaboration of MEV based on CAD models. In this research work, a knowledge base is proposed, which is driven by hierarchical ontologies.

5.3.1 Architecture of MEV Ontologies

In OntoCAD, annotation data is classified into overlapping MEV controlled by OKB. The OKB shown in Figure 34 consists of three layered ontologies: Foundation Ontology (FO), Engineering Viewpoint Ontologies (EVOs) and Application Ontologies (AOs). Each layer

contains Tbox and Rbox statements for terms and Abox statements for assertions with instances.

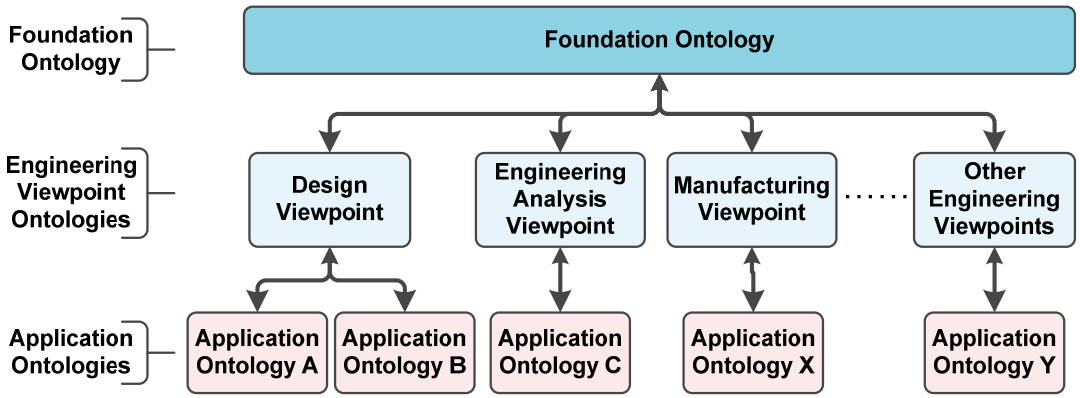


Figure 34 Three Layered Ontology Architecture of OntoCAD Knowledge Base

Foundation Ontology

The top layer is the FO, where common knowledge is defined including data types (Table 19), measurement units (Table 20) and geometric representation (Table 21). In Table 19, the sections in gray are not implemented, as they are either duplication to OWL classes or OWL classes that can be defined using concerned properties, which belongs to the domain of EVO and will be introduced in the forthcoming section.

Measurement units can be mainly classified as si_unit, conversion_based_unit, and context_dependent_unit, in which the si_unit refers to the International System of Units (SI), defined conforming to ISO 10303-41:1994 (ISO 1994e) as shown in Table 20. These SI units can also be classified in terms of their corresponding applications, including length_unit, mass_unit, plane_angle_unit, solid_angle_unit, area_unit, and volume_unit.

Apart from data types and measurement units, the most important function of the FO is to provide description ability for the forms of artefacts. The entity advanced_brep_shape_representation defined in ISO 10303-203:1994 is a type of shape representation that mainly conforms to Class 6 geometric representation that includes Class 1 and shapes represented by advanced B-rep. Geometries can be constructed with instances of class representation_item to cover the three levels of granularity: manifold_solid_brep, advanced_face and edge_curve (Table 18). There are also other representation items, such as axis2_placement_3d, face_bound, point, vertex that potentially support further extension. For complete list of representation_item please refer to Appendix 1 or ISO 10303-203:1994.

Table 19 Data Types Conforming to ISO 10303-21:1994 and ISO 10303-11:1994

Groups	Data Types	Description and Examples
Simple data types	Real	Real numbers, e.g. “1.2345”
	Integer	Integer numbers, e.g. “12345”
	Logical	Three literals: T (true), F (false) and U (unknown)
	Boolean	Two literals: T (true) and F (false)
	String	Sequences of characters, e.g. “This is a string.”
	Binary	Sequences of bits ‘0’ or ‘1’, e.g. “0101 0101”
Named data types	Entity data	It is ENTITY declaration and duplicate form to OWL classes.
	Defined data	It is TYPE declaration, duplicate form to OWL class defined by “owl: equivalentClass”.
Constructed data types	Enumeration data	An ordered set of names, where the names represent values, e.g. “.STEEL”. It is duplicate form to OWL enumerated class “owl: oneOf”.
	Select data	It is duplicate form to structured declarations of OWL classes.
Aggregation data types	Array	A fixed-size ordered collection, e.g. the coordinates for vertexes of faces in a tetrahedron can be: ((a [0, 0, 0], b [1, 0, 0], c [0, 1, 0]) … (d [0, 0, 1], b [1, 0, 0], c [0, 1, 0]))
	List	A sequence of elements that are accessible by their positions, e.g. a list of string: (‘hello’, ‘world’)
	Bag	An unordered collection that may have duplicates, e.g. a set of identical fasteners in an assembly.
	Set	An unordered collection of elements with no duplicates, e.g. coordinates for vertexes of a face: (a [0, 0, 0], b [1, 0, 0], c [0, 1, 0])

Table 20 The Components of OWL Class si_unit Conforming to ISO 10303-203:1994

Components of Class SI Units	Enumerated Classes
si_prefix	exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto
si_unit_name	metre, gram, second, ampere, kelvin, mole, candela, radian, steradian, hertz, newton, pascal, joule, watt, coulomb, volt, farad, ohm, siemens, weber, tesla, henry, degree_celsius, lumen, lux, becquerel, gray, sievert.

For all anchor types, the legitimate association are defined by using axioms constructed from OWL properties with ranges and domains. The association directly related to geometric models are as shown in Table 21. The indirect association are as defined as the axioms of the classes. The associations are accumulated along the growth of the

ontologies.

Table 21 Legitimate Association Defined for Geometric Anchors

Annotation Anchor Granularity	Annotation Anchor (OWL Class)	Examples of Legitimate Association
G1	manifold_solid_brep	Manufacturing process, material, weight and global element size.
G2	advanced_face	Manufacturing process, material, displacement, and load.
G3	edge_curve	Displacement and pressure.

Engineering Viewpoint Ontology

The second layer is the aggregation of EVOs that represent ontologies for EVs. An EVO is a collection of facets (classes) to represent a concept in a domain of interest. And there can be some overlap among EVOs or interconnections among classes across EVOs as illustrated in Figure 35. Therefore each EVO may be treated as an integral collection of partial views in other EVs. For instance, material itself can be treated as a primary EV, which can be part of a design EVO (Figure 35), part of a manufacturing EVO, and also referenced by an analysis EVO. This ultimately constructs interlaced taxonomies.

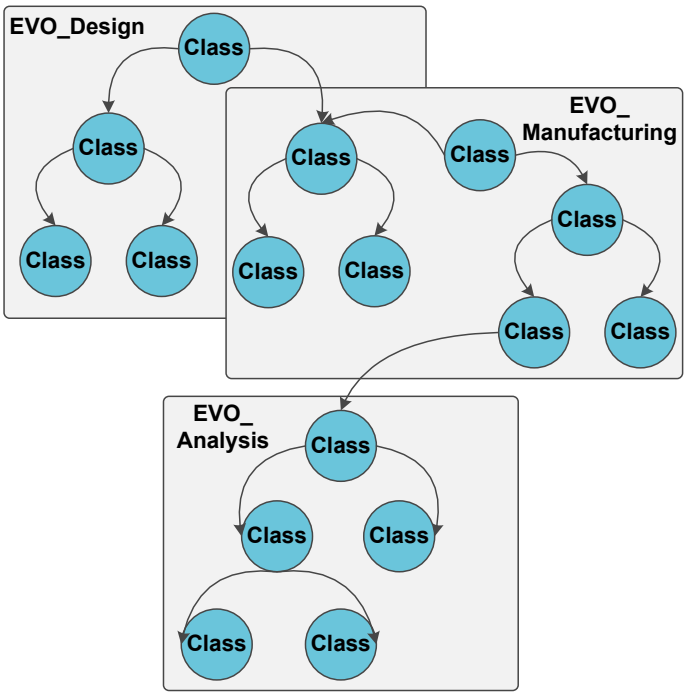


Figure 35 Overlapped or Referenced Classes across EVOs

Application Ontology

The third layer is a collection of AOs, which are basically vocabularies that define the terms and properties used in specific applications. Terms and their inter-relationships for a specific application to be functional are represented as OWL entities (classes and individuals) in an AO. Each defined class in an AO must associate with an EVO or the FO, and no classes in any AO can be isolated. This will make sure all annotations are related and ultimately referenced to the primary anchors. It also ensures that knowledge can be traversed and propagated through the entire knowledge base. How knowledge can be conveyed will be described in the following section.

Once an AO is constructed, the corresponding application with an interface to the OMA is thus able to provide service based on a primary CAD system. In other words, an external engineering application is integrated with the OntoCAD system as a whole. Not only the AOs, but also the FO and EVOs are all extendable and evolvable by incorporating more primary concepts, EVs or applications over time. However, to completely cover all engineering domains implies extensive work, which is out of the scope in this research. Instead, it is tried to explore and define a general systematic mechanism to represent and manage knowledge, therefore to aid engineering design by incorporating MEV.

5.3.2 Knowledge Sharing and Exchange through MEV Ontologies

Knowledge sharing implies that knowledge as a common piece of semantic information is referenced by internal agents (computer or human), while exchange implies that metadata representing knowledge are passed or copied either automatically or interactively with external environments (Jasper and Uschold 1999). This layered architecture supports both sharing and exchanging. Firstly, it enables knowledge transformation, namely codifies tacit knowledge (e.g. design intention and experience) into explicit knowledge (e.g. metadata). The knowledge can then be shared both horizontally and vertically within the OKB. Horizontal dimension refers to knowledge being conveyed within a single level. For example, the knowledge in the EVO for manufacturing can be used by the EVO for cost analysis. Vertical dimension refers to knowledge being shared throughout the three layers. Since all classes are mapped to their superior classes and ultimately all refer to the primary resource, knowledge can be propagated from one end to the other. This is due to the fact that knowledge at ontology at layer ‘n+1’ can be understood by ontology at layer ‘n’ (Jasper and Uschold 1999).

Considering a scenario, the “Application Ontology C” in Figure 34 can be defined for a

cost application and “Application Ontology X” is for a CAM application, and both are at layer L_2 - AO. If a product design – for example a car tow bar - requires a fixture, then it implies the geometric model should include holes at its base, which are instantiated as anchors defined in the FO – the layer L_0. This knowledge serves the CAM application to plan a drilling manufacturing process through manufacturing EVO – the layer L_1. On the other hand, if the cost tool needs the knowledge of the manufacturing process plan in order to set up a costing project and to calculate the costs, the OMA on behalf of application C makes a query to the application X through the route from “Application Ontology C” (L_2), cost analysis EVO (L_1), Foundation Ontology (L_0), manufacturing EVO (L_1) to “Application Ontology X” (L_2) as illustrated in Figure 36. Therefore, the formally coded MEV knowledge can be communicated within the OKB, and exchanged with external environment (engineering applications) through the OMA.

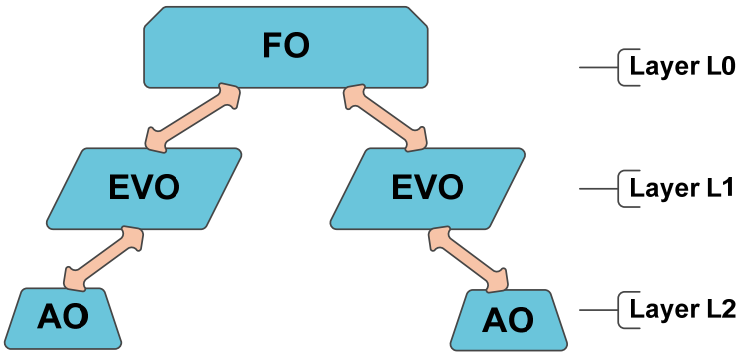


Figure 36 Knowledge Propagation Based on the Three-Layered MEV Ontology Architecture

5.4 OntoCAD Knowledge Modelling Methodology

Since the EVO and AO are modularized units to the knowledge base, they enable new knowledge or concepts to be readily integrated as a total system. To formalise and smooth such a process, a general ontology modelling methodology is required.

Depending on different situations, two modelling strategies are combined: middle-out and bottom-up (Figure 37). In the case of modelling FO from scratch or adding new EVO for general use without a particular downstream application, a middle-out approach METHONTOLOGY is adapted. On the other hand, if it is about integrating a particular application into existing ontologies rather than from scratch, a bottom-up strategy KACTUS can be adapted. The bottom-up strategy is embedded into the middle-out approach as the processes in the late stages are similar, including implementation, evaluation and documentation. The two original strategies have been briefly introduced in Section 4.2.6. The following subsections will described the adapted methodologies in more detail.

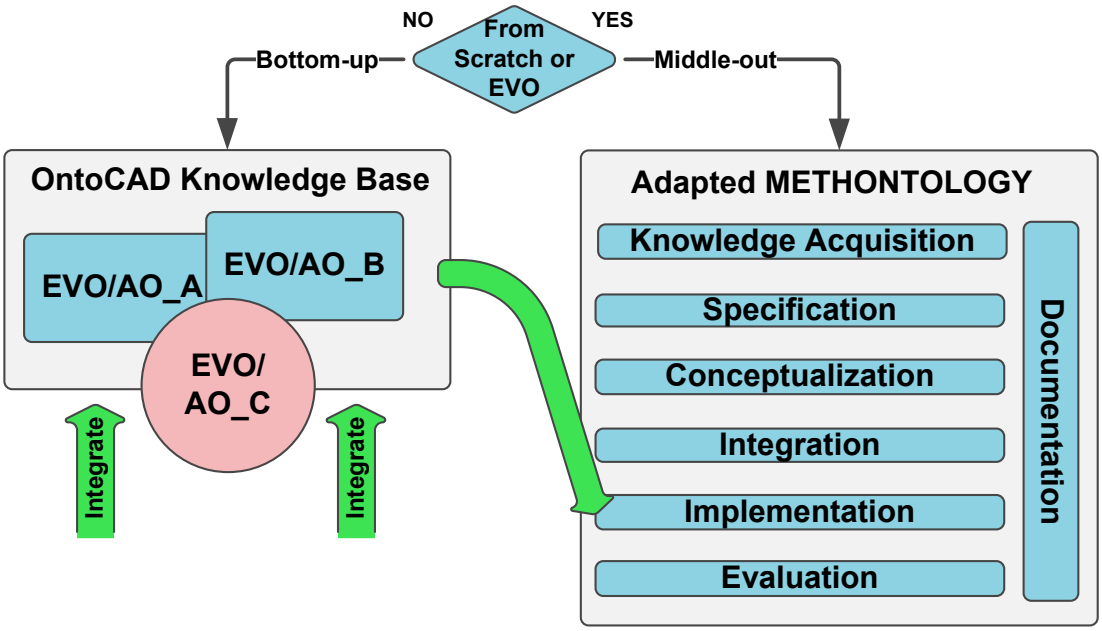


Figure 37 OntoCAD Ontology Modelling Methodology

5.4.1 The Middle-Out Strategy

As noted, a middle-out strategy identifies the most important concepts (e.g. concepts in an EVO) first and then generalizes and specializes into other concepts, which are from AO/EV upwards FO, or from FO/EV downwards AO in this present work. The customized METHONTOLOGY consists of a series of processes: knowledge acquisition, specification, conceptualization, integration, implementation, evaluation and documentation. It should be noted, that these processes (e.g. knowledge acquisition here) are different from the knowledge processes that try to be addressed by the proposed OntoCAD system. It specifically refers to the processes during ontology modelling.

Knowledge Acquisition

This phase is often a simultaneous process to the next phase – the specification. It aims to provide initial support for ontology modellers to construct ontology specification documents by various methods as shown in Figure 38. For example, if the interested domain is not clear, the ontology modellers can define the scope and purpose through interviews with experts, brainstorming and literature research in books, handbooks, figures, tables, existing ontologies etc. From left to right illustrated in Figure 38, the study methods gradually evolve from informal to more formal and the concepts in the interested domain become more explicit.

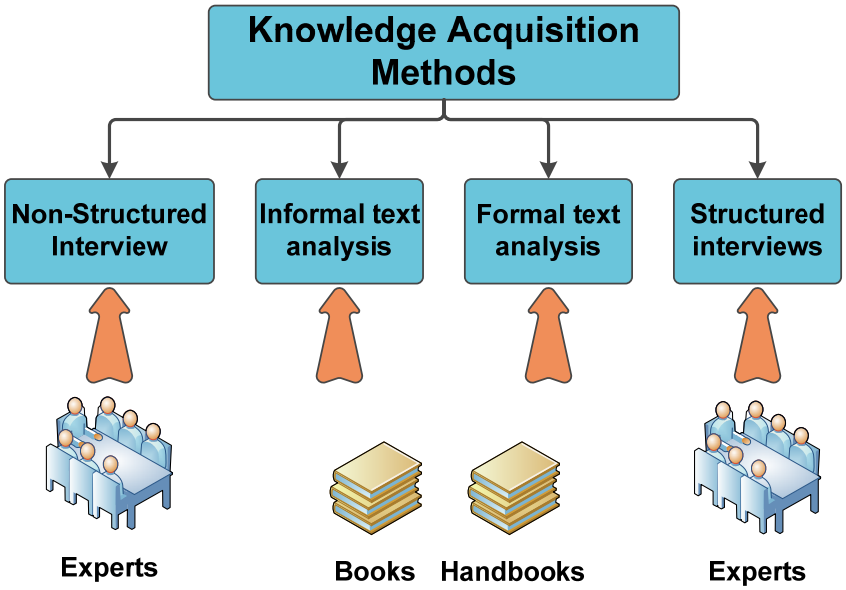


Figure 38 Knowledge Acquisition Methods

Specification

The objectives in this phase are to produce an ontology requirement specification document that semi-formally (intermediate representations) defines the information: the purpose of the ontology; the scope that includes the set of terms, characteristics and granularity; and document control. An exemplar document is shown in Table 22.

Table 22 Ontology Requirement Specification Document in the EV of Manufacturing

Ontology Requirement Specification Document	
Domain	Manufacturing
Date	24/07/2011
Conceptualized by	Chunlei Li
Implemented by	Chunlei Li
Purpose	Ontology about manufacturing processes, in which information including process description, applicable material, process variations, economic considerations, typical applications, design aspects and quality issues.
Scope	A list of manufacturing process categories: casting processes, plastic and composite processing, forming processes, machining processes, non-traditional machining processes, and fabrication and joining processes. In each category, there are a number of manufacturing processes. And each process has their corresponding characteristics.
Source of Knowledge	K.G. Swift, J.D. Booker, Process Selection: from design to manufacture, Elsevier Butterworth-Heinemann, 2003.

Conceptualization

In the conceptualization phase, domain knowledge is structured into a conceptual model – a domain vocabulary that describes problems and solutions concluded in previous phases. In general, this phase needs four tasks as shown in Figure 39.

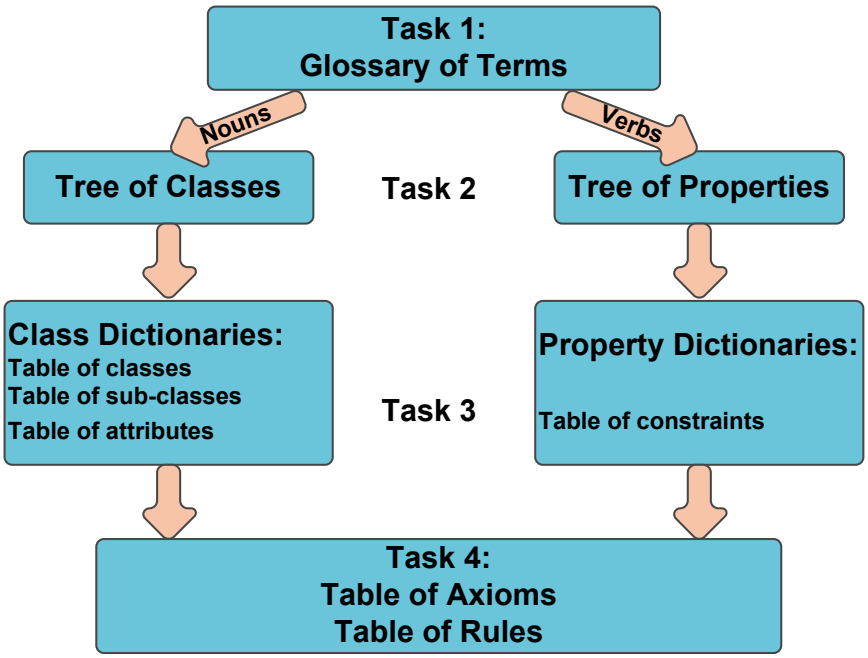


Figure 39 The Tasks in Ontology Conceptualization Phase

Task 1. To build the glossary of terms (GT). The GT contains all concerned terms of the domain of interest, including concepts, instances, attributes, relations between concepts, and between concepts and their attributes. This table also contains the natural language descriptions and their synonyms and acronyms, as depicted in Table 23.

Table 23 Example of the glossary of terms (GT)

Terms	Description	Synonyms	Acronyms	Entity Types
Sand casting	Moist bonded sand is packed around a pattern. The pattern is removed and molten metal poured into the cavity. Risers supply necessary molten material during solidification. The mould is broken to remove the part.			Class
Acrylonitrile butadiene styrene	A type of common thermoplastic (chemical formula (C8H8)x· (C4H6)y·(C3H3N)z).		ABS	Class

Task 2. To build taxonomies with names and structures of classes and properties at a high level of abstraction. This task has two routes: the classes and the properties.

The classes are derived from nouns from the specification document, while the properties are the relations represented by verbs. Examples are given in Table 23 and Table 24 respectively.

Table 24 Examples of Class Taxonomy

Class	Super-classes	Sub-Classes
Casting	Manufacturing process	Sand casting; Evaporative pattern casting.
Evaporative pattern casting	Casting	Full mould casting; Lost foam casting.

Table 25 Examples of Property Taxonomy

Property	Type	Super-property	Sub-property
Has manufacturing process	Object	N/A	N/A
Has weight value	Data	Has value	N/A

Task 3. To build two dictionaries along the two routes: a) class dictionaries and b) property dictionaries. In route a), having the classes and properties defined in previous tasks, classes are able to be more explicitly described by relating classes, subclasses and their attributes with appropriate properties, as illustrated in Table 26. In the route b), properties, sub-properties and their constraints including ranges and domains are able to be defined as needed in defining class dictionaries, as illustrated in Table 27.

Table 26 An Example of Class Dictionaries

Class	Class Attributes	Individual Attributes	Properties
Sand casting	Description	N/A	Has description
	Material	N/A	Has material
Material Density	Description	N/A	Has description
	Value	N/A	Has value
	Unit	g/m2	Has unit

Table 27 An Example of the Property Dictionaries

Property	Domain	Range	Cardinality	Inverse Property
Has material	Manufacturing process	Material	1:N	Is material of
Has unit	Material density	Unit	1:1	Is unit of

Task 4. The objectives in this task are to synthesize the classes and properties in fine detail including necessary attributes, to define constants, to define a table of formal

axioms and rules, and finally to instantiate individuals if necessary. In order to further refine the synthesized dictionaries, the characteristics of classes and properties are also defined here and appended to Table 26 and Table 27 with the available options as listed in Table 28.

Table 28 Types of Relations Used To Build Taxonomies

Relation Characteristics	Descriptions	Apply to
Disjoint	Distinct from each other, e.g. manufacturing process extrusion is a class disjoint from casting.	Class
Inverse	Relation in reverse, e.g. “has parent” vs. “has child”.	Object property
Functional	For a given individual, there can be at most one individual that is related to the individual via the property.	Object property Data property
Inverse functional	It refers to the inverse property is functional.	Object property
Transitive	If individual ‘a’ is related to individual ‘b’, and also individual ‘b’ is related to individual ‘c’, then individual ‘a’ is related to individual ‘c’ via property ‘P’.	Object property
Symmetric	The property relates individual ‘a’ to individual ‘b’ then individual ‘b’ is also related to individual ‘a’ via property ‘P’.	Object property
Anti-symmetric	The property relates individual ‘a’ to individual ‘b’ then individual ‘b’ cannot be related to individual ‘a’ via property ‘P’.	Object property
Reflexive	A property ‘P’ is said to be reflexive when the property must relate individual ‘a’ to itself.	Object property
Irreflexive	A property relates an individual ‘a’ to individual ‘b’, where individual ‘a’ and individual ‘b’ are not the same.	Object property

Another objective here is to define engineering rules based on the yet informal ontology. An axiom example for an engineering constraint (Table 29) implies only a part that has weight range from 200 grams to 100 kilograms is applicable to the manufacturing process sand casting.

Table 29 Example of Sand Casting Rule on Size Ranges

Axiom Name	Sand Casting Size Range Rule
Description	The size range for manufacturing process sand casting can be applied from 200 grams to 100 kilograms in weight.
Expression[9]	If part(?p) ^ hasManufacturingProcess(?part, SandCasting) ^ hasWeight(?p, ?w) ^ hasValue(?w, ?v) ^ hasUnit(?w, ?u) ^ swrlb:greaterThan(?v, 0.2) ^ swrlb:lessThan(?v, 100) Then isLegal (?B_sand_casting_size_range_rule, TRUE)

Integration

Integration aims to reuse existing ontologies or parts of them if the documents or/and the ontologies are available. Having constructed these semi-formal specifications in previous phases, ontology modellers are able to do two inspections: a) inspect the existing documents and meta-ontologies if available, and b) inspect the meta-ontologies only if documents are not available. If any classes and/or properties are reusable, they should be referenced and updated in the documents at conceptualization phase and recorded in an integration document, as illustrated in Table 30.

Table 30 Example of Integration Document

New Classes	**Reusable Source Ontology**		**Modification**
	Ontologies	**Classes/Properties**	
gram	SI_unit	gram	
kilogram	N/A	N/A	Update to derived_unit and refer to SI_unit.

Implementation

This phase aims to codify the semi-formal documents developed earlier into formal ontology data, including the GT, the dictionaries and rules. The implementation or codification of ontologies requires ontology specification languages and modelling tools, which are introduced in Section 4.2. As noted, the chosen language is OWL-DL and the ontology editor is Protégé in this work.

[9] To be noted that SWRL and SQWRL notations and conventions are used for illustration purpose only. It is not absolutely necessary in the implementation.

Evaluation

The evaluation phase is composed of two tasks: verification and validation. An evaluation plan should be defined and documented for the evaluation process to follow and for future reference, which is called functionality acceptance test (FAT) (See Appendix 2).

The verification (debugging) refers to testifying whether the ontology is correctly implemented. It mainly and firstly checks whether the ontology is coded conforming to its specification language schema, which is often enforced by the modelling tools. Secondly it also checks the consistency so that there are no contradictions in the defined axioms and rules. This largely can rely on the reasoner provided by many ontology modelling tools.

Validation refers to the technical judgement that ensures the correctness of the ontologies reflecting to users' need. This refers firstly to evaluating whether the requirement specification documents, the GT and its tables of axioms and rules are correctly composed, including the logic correctness, completeness and redundancy. Secondly, it refers to evaluating whether the ontologies developed conform to these documents, to ensuring the correct thing is built to provide the needed service.

Documentation

The documentation is carried out simultaneously throughout all development processes. Each process is documented as listed in Table 31.

Table 31 The Documents Used During the Ontology Modelling Process

Middle-Out Ontology Modelling Phases	Documents
Knowledge acquisition	● Meeting minutes ● Reading lists ● Questionnaires ● Interview forms
Specification	Ontology requirement specification documents.
Conceptualization	● The GT document ● Taxonomy document ● Table of classes and table of properties ● Synthesized informal ontologies and rules
Integration	Integration document
Implementation	The ontology data and codification logbook
Evaluation	Evaluation plan – FAT

In these documents, document control is an important part, which contains information including the document identifier with version numbers, file number, status, editors, authorisers, change history, retention period and so on. A document control section complied with ISO 9001:2008 (ISO 2008a) is defined and appended to the end of each document (see Appendix 2).

5.4.2 The Bottom-Up Strategy

Differing from the middle-out strategy, the bottom-up strategy identifies the most specific concepts first according to the ontology being integrated and then generalized into more abstract concepts. Since the later phases are identical to the middle-out approach, they are merged at the end.

In the case of integrating a specific application into an existing knowledge base as a total system, the complete list of terms are already known and each of them needs to be integrated, regardless of whether corresponding EVOs are available or not. Therefore the bottom-up strategy is naturally suitable for describing and integrating a known application into existing ontologies rather than developing from scratch. It is theoretically easier to be initialized, and the ontology can be developed more quickly than the middle-out strategy, which is evaluated in Section 7.4. Further more, not only the AOs, in the case of upper level ontology EVO is not defined, a corresponding EVO can also be derived along the development.

The first four processes in the middle-out strategy are substituted by the first two processes in the original KACTUS: specification of the application and integration as shown in Figure 40.

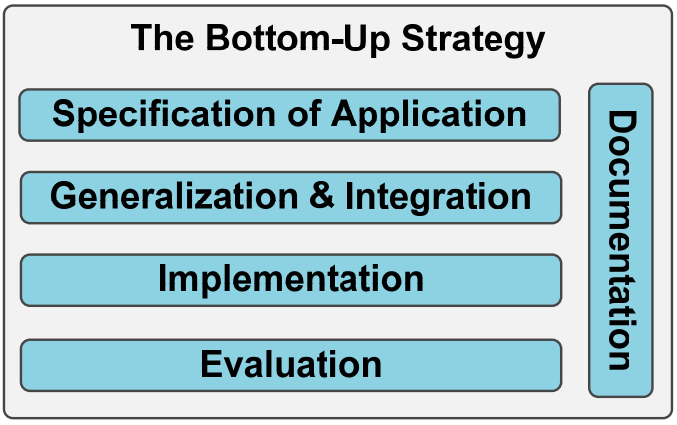


Figure 40 Tasks in the Bottom-Up Strategy

Specification of the application

In this first process, the knowledge acquisition and specification in the middle-out strategy can be skipped as the terms in an application are relatively straightforward to answer (e.g. explicitly defined in a software program or a manual book), rather than acquired from a preliminary study. Terms can be retrieved from the application itself such as help manual or from user handbook. Therefore, the guidelines for conceptualization phase in middle-out strategy can be followed, including developing the GT document, and documents for tables of classes, properties, synthesis and rules (as illustrated in Table 23, Table 24, Table 25, Table 26, Table 27 and Table 29). The formats for these documents can be identical. The preliminary generalization on the terms is carried out when developing taxonomies.

Generalization and Integration

The integration process is different from the integration phase in the middle-out strategy. The main purpose in the middle-out strategy is to find a reusable ontology or its components, while the main purpose in the bottom-up strategy tries to map the application terms into the existing ontologies, extend the exiting ontology by patches, or generalize a new EVO or components in the FO whichever is appropriate. Although the purpose varies, the integration document can be shared as depicted in Table 30.

The most complex case can be where the corresponding EVO does not exist at all. In this case, the generalization can refer to a simplified preliminary study in the bottom-up study. The application specific terms need to be generalized through knowledge acquisition and ontology requirement specification process, where the corresponding documents such as Table 22 should be correctly recorded.

Merging Point with the Middle-Out Strategy

Apart from the first two phases in the bottom-up strategy, the rest of the phases – the implementation and evaluation phases can be commonly shared including the document formats and document control mechanism. In the same way as the middle-out strategy, the documentation process should always be simultaneously carried out during the entire modelling process.

5.5 Reasoning

Having described how knowledge can be acquired (i.e. OGUI), represented (i.e. annotation data structure), managed (i.e. MEV ontology architecture), modelled (i.e. OntoCAD ontology modelling methodology), and used as basis for consistent systems (i.e. interoperability and system integration), we now look at how knowledge can be further used, namely reasoning, which is the key to the automation of knowledge processing.

As described before, OMA plays the important role of reasoning. Reasoning is based on the ontologies with the instantiated individuals in the OKB, and reasoning over the OKB decides the knowledge reusability and evolvability. The reasoning activities can be carried out by the reasoners provided by most recently developed ontology modelling tools. The research in algorithms or developing reasoners is beyond the scope of this present work. Instead, we focus on how to effectively use the reasoner (classifier) to aid the engineering design process.

Based on OWL-DL, two standard reasoning services can be provided: classification and consistency checking (Drummond et al. 2009). Classification can be further divided into two types: class subsumption and individual membership (Jasper and Uschold 1999). Class subsumption is to check whether one class is a subclass or the equivalent of another, in which way a subsumption hierarchy (taxonomy) can be computed. Individual membership is to check whether an individual is an instance of a class. Consistency checking is to test whether a class can be instantiated based on its conditions without any contradiction. A class is inconsistent if it can not be instantiated.

On the other hand, in terms of levels of knowledge abstraction, knowledge can be classified into factual (i.e. description of data), conceptual (i.e. classes) and methodological knowledge (i.e. ontology languages and models) (Andrea et al. 2008). Analogous to this classification, reasoning services offered by OntoCAD can be categorised into three types (factual, conceptual and methodological reasoning), and coordinated by OMA.

5.5.1 Factual reasoning

Factual reasoning mainly refers to all reasoning tasks operating at the data level. The appliance of factual reasoning includes the consistency checking on individuals, individual membership and data query.

Consistency checking on individuals ensures the ontology metadata is legitimate and

individuals are properly instantiated during the development of OKB ontologies (e.g. debugging ontology data) and annotating processes. For example, if “ABS” is instantiated as an individual of “Thermoplastic”, and also assigned as type of “Iron” which the latter is explicitly defined distinct from “Thermoplastic”, the reasoner is able to perceive the conflict and give explanations. This verification facility is very helpful during ontology development.

As noted, individual membership computes the ownership of individuals, namely check whether a given individual is an instance of a class, or whether a class has been satisfied to have individuals. With this reasoning task, geometric topology can be perceived through the transitive properties. For example, to testify that anchor “EDGE_1” is part of “BODY_1” due to the fact that the parent ‘FACE_1’ is part of ‘BODY_1’.

Furthermore, individual membership makes the use of rules for engineering experience and constraints through quantifier restrictions and value partitions, namely the effects of changes in quantity. Value partitions restrict the range of possible values to an exhaustive list, for example, a specific number of product quantity can be categorized as either “MassProduction” or “Prototype” appropriately according to the definition of “ProductionScalePartition” in the ontology. Thus as any individual falls in different categories, further corresponding manufacturing rules may be applied.

Another intensive use of individual membership is the application watchdog. An AW is in fact a named class with conditions representing rules that are constructed from axioms. An OMA reasoner checks whether this class (AW) is satisfied with any individual on the event of ontology changes. As a consequence, the status of an ontology model can be monitored. For example, an AW can monitor whether a set of necessary parameters that serves a cost analysis becomes available, such as production quantity, manufacturing processes, mass and materials. Or an AW can be assigned to watch a particular engineering constraint, therefore bringing this users’ attention for further process, such as to give real-time advice about manufacturing processes based on the changes of minimum wall thickness or number of thru holes.

5.5.2 Conceptual reasoning

Conceptual reasoning is to reason over the conceptual level of knowledge, which is based on the standard reasoning services – class subsumption including equivalency checking. This service enables the system to be context aware, based on the inference over class conditions. Similar to factual reasoning, conceptual reasoning can be used for

consistency checking on classes.

Another appliance of conceptual reasoning is to configure an intuitive interface for users or computer applications to collect data and knowledge. Expert knowledge can be defined and enforces non-expert users to comply with the constraints. For example, if users claim themselves as cost engineers, then the GUI in the OGUI should be cost oriented. Therefore, all cost related annotation options should be available, and this is the inferred class hierarchy according to defined EVO. As the example shown in Figure 41, "CostDriver", as a subclass of EVO "EVO_Cost", are defined as any class that affects cost (necessary and sufficient conditions), and subclasses of other EVOs "ManufacturingProcess" and "Material" also contain this axiom (necessary conditions only). Having processed by an OMA reasoner, it can be deduced that "ManufacturingProcess", "Material" and all their own subclasses can be necessarily subclasses of "CostDriver".

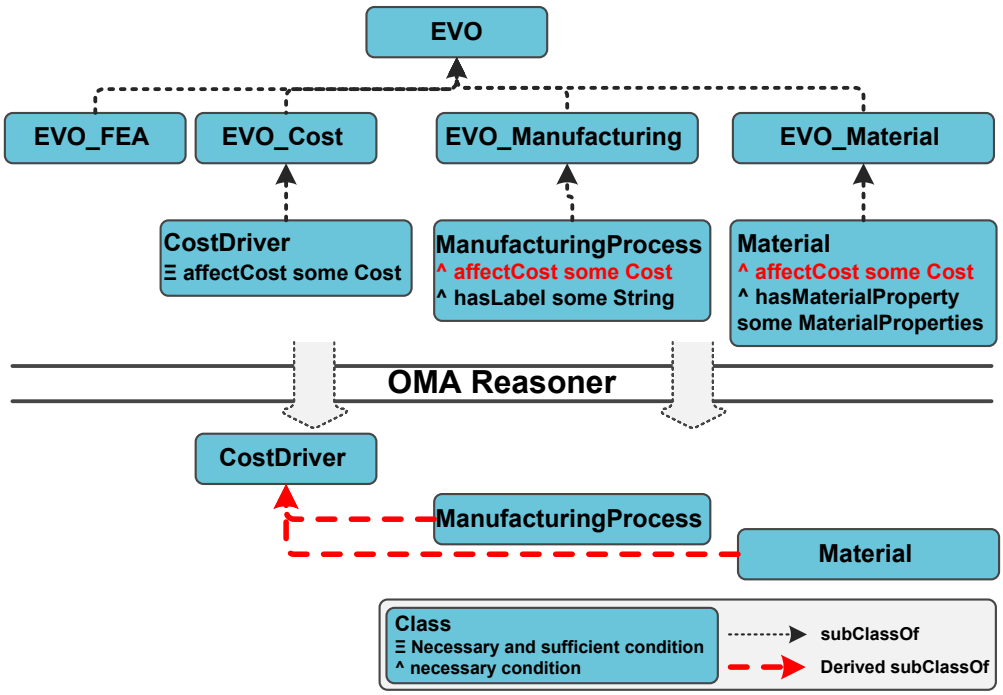


Figure 41 Example of Conceptual Reasoning

Furthermore, conceptual reasoning enforces end users to comply with the constraints of anchors. As noted that all information/knowledge is rooted in geometric models and the legitimate association is defined in the OKB ontologies (Table 21), the behaviours of engineers or computer agents are restricted accordingly. For example, when a G2 anchor (face) is selected, weight is not available for annotating, but becomes available if a G1 anchor (solid body) is selected. Therefore, the bridges across FO (containing geometric model) and the other two lower levels of ontologies – EVOs and AOs can be established.

5.5.3 Methodological reasoning

Methodological reasoning refers to dynamically deducing a result over both data and conceptual levels of knowledge, and maybe across ontologies. One appliance of methodological reasoning is semantic data query, which differs from standard data query in the mean of involving reasoning on class conditions and ontology interrelations. Semantic query here refers to accurate and explicit data or class retrieval according to its context from OKB, such as using the synonyms of the ontological vocabulary. For example, the AO term “Powdered_Metals” (a subclass of “PRODUCT_DESCRIPTION_-_Process”) defined for a costing tool SEER-MFG is equivalent to EVO term “PowderMetallurgyMolding” (a subclass of “ManufacturingProcess”). When querying an instance of manufacturing process “Powdered_Metals”, instances of its synonyms will be also retrieved. This mechanism builds the bridges among EVO and AO.

Methodological reasoning can also be used for defining engineering constraints through the cooperation between EVOs, such as engineering rules for manufacturing and cost engineering. For example, methodological reasoning can inform users that the only option to manufacture a particular product is sand casting if the material is an instance of aluminium (conceptual reasoning on class conditions) while die casting and sand casting are the only available casting processes within a particular manufacturing factory (factual reasoning). For another example, a constraint on manufacturing process sand casting can be the size of a product range from 20 grams to 400 tonnes in weight (Swift and Booker 2003). In practice, this range can be reduced by particular companies to correspond to their manufacturing capability, such as from 200 grams to 100 kilograms. Based on a manufacturing viewpoint, if a model is designed with sand casting but out of this range in weight, it will be identified and suggestions can be made for manufactors. Or in the case of the weight is known, a judgement can be made if sand casting is a candidate manufacturing process.

Moreover, the expressiveness and semantics can be extended by rule languages. With the SWRL rules, mathematical relations can be understood. As a result, conceptual reasoning can test for equivalency of classes, and in turn to assist with semantic data query. For examples, equivalent classes with necessary and sufficient conditions can be defined as one kilogram is equivalent to 1,000 grams or 2.2 pounds, or the mass of a solid body equals its material density times its volume. In a particular case, the value of an individual “weight_body_01” is recorded as 1.9852 in kilogram. On a specific query with a particular constraint on measurement unit, the value can be automatically converted to

4.3766 in pounds and returned. The OMA uses such sophisticated rules for complex reasoning to improve the level of process automation. As a fact, this is the feature of procedural annotation.

Within the OntoCAD system, all OntoCAD key modules are affected by reasoning activities. These three types of reasoning activities are driven by the OKB module and executed by an ontology reasoner belonging to the OMA. Therefore, the OMA processes and reuses the knowledge to serve other modules to aid in engineering design process. On the other hand, reasoning activities also assist with the development of the OKB. It includes debugging ontologies, avoiding the redundancy during ontology integration, discovering equivalent descriptions, reusing and refining concept descriptions. As a consequence, the derived knowledge by reasoning activities can be patched to the OKB so that reasoning rules are reusable and keep the knowledge base evolvable and extendable over time.

5.6 Concluding Remarks

According to the previous study, it has been identified that annotation and ontological technologies can be combined with CAD systems to improve knowledge and information management. Based on these findings and inspired by two engineering cases, a general purpose framework called OntoCAD (Ontology-driven semantic annotation framework for CAD systems) is proposed to assist with engineering processes by incorporating multiple engineering viewpoints.

Within this framework, a basic OntoCAD annotation data structure is proposed, on which a standard-compliant annotation anchoring mechanism is defined based. Annotations are used to semantically re-represent the B-rep geometric models, to record viewpoint-dependent information associated with the models, such as manufacturing process and costing data, structural loads and constraints etc.

OntoCAD is designed based on the annotation data structure and consists of three key modules: OntoCAD Graphical User Interface (OGUI), OntoCAD Knowledge Base (OKB) and OntoCAD MEV Agent (OMA). The module of OGUI is embedded in a CAD system and is responsible for interacting with end users to capture inputs. The OKB is a repository of ontologies that represent the semantics in a three-layered architecture, in order to manage expertise as multiple engineering viewpoints. The module of OMA is an intelligent broker who aids the synergy among CAD system, the knowledge base and external engineering tool or services.

Since ontologies are used to establish a consistent, extendable knowledge base and annotation plays the role of information media, an important feature – reasoning services - is provided, including knowledge base consistency checking, semantic data query and methodological reasoning.

Moreover, in order to aid formally establish and maintain the OKB, an OntoCAD knowledge modelling methodology is derived by combining a middle-out strategy and a bottom-up strategy. The first one suits for developing the foundation ontology or an engineering viewpoint ontology from scratch. And the later strategy suits for integrating an engineering viewpoint ontology into existing knowledge base or modelling an application ontology.

In a nutshell, this chapter has introduced an intended support for knowledge and information management based on CAD systems in order to aid engineering design. The impacts of each computational enablers employed in the proposed OntoCAD system is illustrated in the impact model – Figure 42.

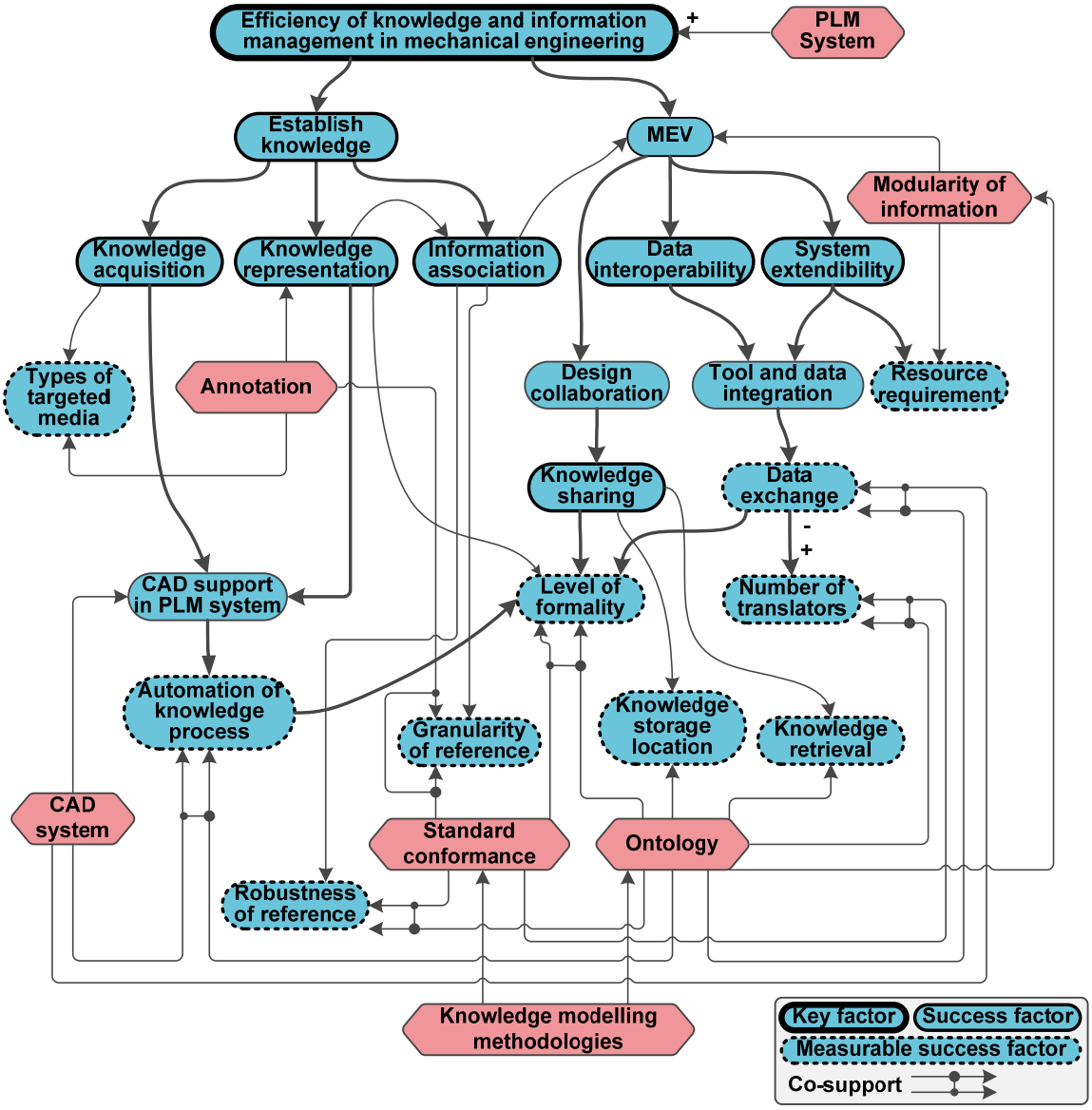


Figure 42 Impact Model in the View of Key Technologies Employed In OntoCAD System

To demonstrate and evaluate this intended support, an actual support – OntoCAD prototype system also has been designed and developed, which operates as a add-on application to a commercial CAD system NX (Siemens PLM Software Inc 2011a). The design and development process will be described in the forthcoming chapter.

Chapter 6 Development of the Actual Support

To improve a situation of a research subject, there may be two types of support: the intended support and the actual support (Blessing and Chakrabarti 2009). The intended support is an ideal support (an envisaged solution) proposed by the research in order to improve the current situation, while the actual support is the realisation of the intended support that is mainly used to evaluate the proposed concept. The actual support may only implement the partial functionalities in a different way to that described in the intended support considering the resource constraints and feasibility. However, the core contribution must be included. The relationship between the intended support, actual support and core contribution are illustrated in Figure 43. In Chapter 5, an intended support for improving the current situation in aid of engineering design process by incorporating MEV in an ontological knowledge base is presented. This chapter will describe an actual support that realizes the concept by developing a software application for demonstrating the feasibility and the usefulness.

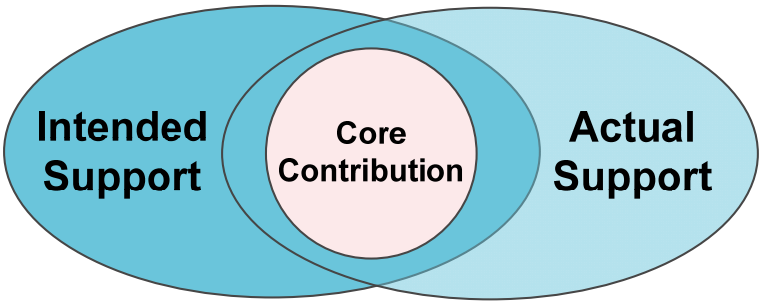


Figure 43 Intended Support and Actual Support (Blessing and Chakrabarti 2009)

In order to do so, the actual support is elaborated by following an iterative software development process (Jalote 2005) as illustrated in Figure 44. Each iteration actually goes through the sequential phases in a waterfall model originally introduced by Royce (1970), namely the requirement specification, design, implementation, verification and maintenance. The reason for choosing this development process is that this research work is an individual project, and the requirements are initially unclear and the design needs to be gradually improved along the progress in research. In contrast, a more contemporary concurrent (team-based) development process is not suitable for an individual project, and a traditional waterfall model does not allow backward changes.

Although the actual support was iteratively refined in the actual experimental work, it will be described in one go as a single process for reasons of conciseness. Therefore, requirements for the actual support will be constructed, and then followed by the design for a prototype of the OntoCAD system, which is actually developed as a JAVA add-on application to a commercial CAD system – Siemens' NX6. An evaluation plan will be

enacted at the end of this chapter and the evaluation process will be carried out and described in Chapter 7, followed by a discussion of the experimental results in Chapter 8.

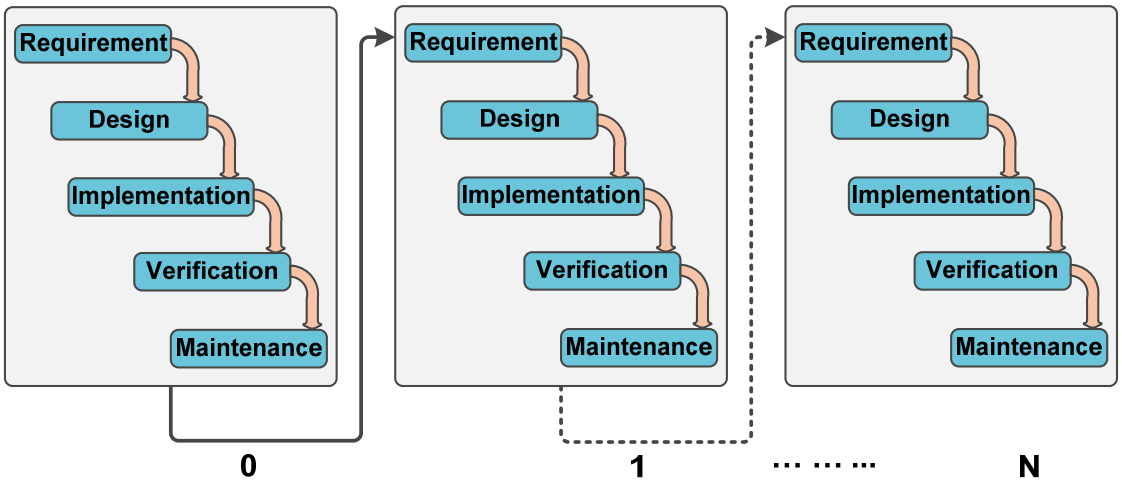


Figure 44 An Iterative Software Development Process (Royce 1970; Jalote 2005)

6.1 Requirement Specifications

Since the core contribution is the essential to demonstrate the applicability and successfulness in an intended support, the requirement specifications will start with a requirement analysis that defines the scope of the core contribution. With some other complementary feasible requirements, a list of requirements for the actual support that differs from the intended support can be defined at this stage.

6.1.1 The Core Contribution

Revisiting the overall structure for the proposed intended support in Chapter 5, the three-module architecture, as depicted in Figure 45 (repeated as Figure 27), is the core contribution that must be realized. The differences between the intended support and the actual support mainly lie in the usability of the GUI, the completeness of knowledge base and the standards-compliance, and the complexity of rules for the MEV agent. Each module will be analyzed in detail.

Requirement Analysis for OntoCAD Graphical User Interface (OGUI)

In the intended support, the OGUI module is expected to address issues for the knowledge acquisition and annotation anchors, the dynamic annotation user interface and rendering system for both human and computer audiences, and the monitoring computer agent – AW. However, with the time constraints and limited manpower, not all of them need to be fully addressed.

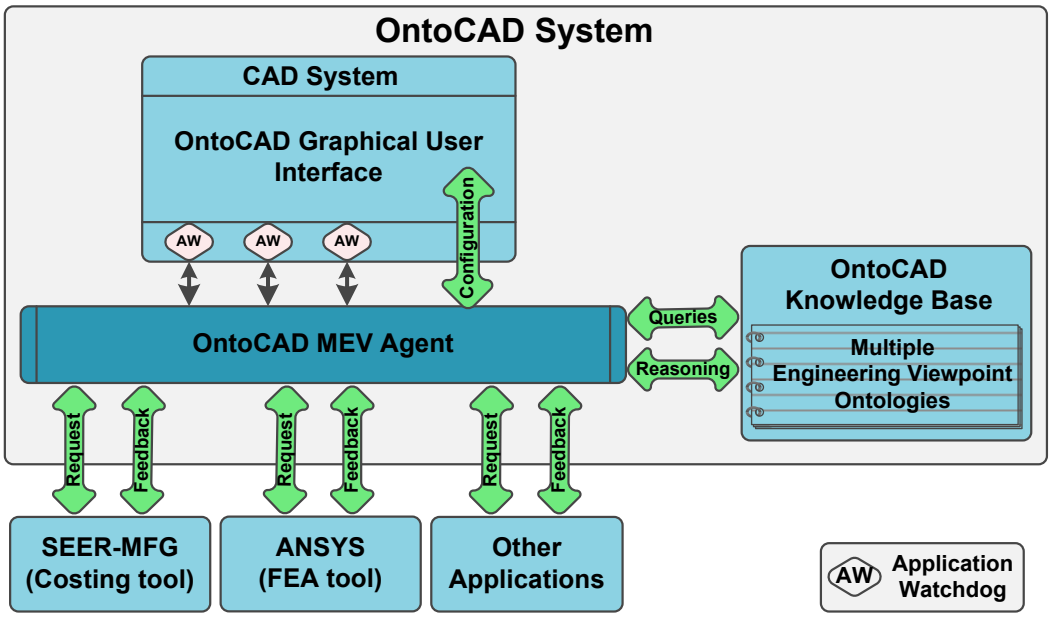


Figure 45 Overview of the OntoCAD System

The function of knowledge acquisition for OGUI must be implemented as it is critical to establishing and evolving the knowledge base. Theoretically, there are two types of knowledge acquisition mechanisms: automatic knowledge acquisition extracted from the CAD system and manual knowledge acquisition from users. CAD related knowledge such as dimensions can be programmatically extracted from CAD systems through APIs. However, what and how CAD related knowledge can be extracted largely depends on the APIs provided by the CAD system vendors. And it is not necessary for this research work to implement all possible extraction. Therefore, only some automation in knowledge acquisition needs to be satisfied for demonstration, such as automatic labelling of entities in the CAD model, extracting information such as volume and mass for a solid body, area for a face. The functionality for manual annotation of that model is more emphasized, such as manually inputting annotations for tolerance and manufacturing processes.

Although any level of granularity for annotation anchoring is theoretically feasible, three levels for anchors are sufficient in the demonstration application, namely edges, faces and bodies. This will be discussed again in Chapter 8. Furthermore, multi-directional anchoring is not a primary goal in the present work.

Importantly, the GUI needs to demonstrate the ability of dynamic adaption reflecting to changes in context (the ontology), including both interfaces for annotating and rendering annotations, which differs from static interface. For example, the interface should be adaptable between object and data annotations, and also adaptable for annotation contents, such as various options of measurement units in data annotations, or data without units such as comments.

The AW is a core functionality that demonstrates how ontological knowledge can assist with design at run-time by handling individual tasks and demonstrating the benefit of reasoning facilities, and so does an interface for data exchange.

Requirement Analysis for OntoCAD Knowledge Base (OKB)

In the intended support, the OKB module is expected to have three key features: stand-off annotation storage, and capability to accommodate knowledge structure and to hold engineering semantics. As concerns annotation data storage, the OKB should support a stand-off strategy to maximise the freedom of data portability and maintenance. The annotation data must support formal knowledge structure as a key feature; however the support for freestyle annotation should not be disregarded. The free annotation feature ensures the extendibility, such as allowing users to add text for general opinions, or establishing references to external information objects through URIs. However, the freestyle annotations have less process-ability. Although this can be improved with the help of technologies such as natural language processing, it is not the primary focus of this project. The OKB module should support both direct and indirect annotations in order to ensure rich semantics, in which the latter one enables chained annotations, as noted in Figure 33 of Chapter 5.

With regard to the semantic features, the OKB should comply with the three-layered architecture of MEV ontologies proposed in Chapter 5, and demonstrate engineering expertise. However, it was decided that this should be specifically developed based on the two case studies (i.e. cost estimation and FEA) introduced in Chapter 3. This is because the proposed solution aims to be able to incorporate as many services/tools as possible as a general purpose framework, but it is not realistic to develop a demonstration application that covers all EVs even only within the mechanical engineering domain.

Requirement Analysis for OntoCAD MEV Agent (OMA)

As noted in Chapter 5, the OKB module in an intended support that is expected to play a role of an intelligent agent that is in charge of coordinating with the other two modules and applications: it interacts with the upstream CAD systems via the OGUI; horizontally, it interacts with the kernel – the OKB; and it interacts with downstream engineering applications, such as a costing tool SEER-MFG.

To interfacing the OGUI, there are two routes: to directly provide ICI for the dynamic GUI

and to handle the AW. The later one implies enacting rules of an AW and assigning them to a specific task. As it is also responsible for interacting with other OntoCAD modules, this may require a complex and sophisticated interface to achieve some level of automation and a satisfactory user experience. However, this requires unaffordable labour resources for development, but can be adequately demonstrated by simplifying the interface or arranging manual set-up. For examples, not all possible data extraction from CAD models will be implemented, but some selected extraction will be necessarily demonstrated. And fully automatic interaction with downstream applications will not be implemented, but a sufficient and generic interface will be provided, since the rest will be more software development rather than research.

6.1.2 The List of Requirements

Based on the aforementioned requirement analysis, a list of requirements can be concluded as recorded in Table 32.

Table 32 Requirement List for the Actual Support

Problem statement (overall aim):	
Define a systematic semantic annotation framework to assist with CAD-based design in the domain of mechanical engineering, where annotations are managed to support MEV (interoperability and extendibility), able to derive a data model and enable downstream processing.	
Intended Support of the OGUI	**Actual Support of the OGUI**
To support both audiences: computer agent and human users.	EQUIVALENT
Fully automatic and manual knowledge acquisition.	Partially automatic and manual knowledge acquisition.
Completion of levels of anchoring granularity including assembly.	Three levels of anchoring granularity.
Dynamically change according to context changes.	EQUIVALENT
To interface AWs and data exchanging.	EQUIVALENT
Intended Support of the OKB	**Actual Support of the OKB**
Stand-off annotations.	EQUIVALENT
To represent both structured and freestyle annotations.	EQUIVALENT
To process both structured and freestyle annotations.	To process structured annotations only.
To support both direct and indirect annotations.	EQUIVALENT
To manage semantic knowledge related to all EVs using the three-layered MEV ontologies.	To demonstrate the use cases of EVs with cost analysis and FEA only.
To handle all types of target media.	To handle CAD models only
Intended Support of the OMA	**Actual Support of the OMA**
To interact with the OGUI, OKB and external applications fully automatically in order to seamlessly integrate downstream applications/services.	To demonstrate reduced set of data extraction from CAD model. To demonstrate interactively with necessary manual set-up.
To provide ICI for dynamic OGUI and handle AW.	EQUIVALENT
To provide sophisticate reasoning ability on all cases.	To demonstrate reasoning with some scenarios in cost analysis and FEA.
Conclusion:	
The intended support describes an ideal solution, while the actual support only demonstrates the core contribution. The compromise is mainly in the means of level of automation, coverage of target media and EVs.	

6.2 Design

In designing the demonstration software application and some other related manual arrangements, two use cases were considered – a cost estimation case and an FEA case. Each case has a different emphasis. The first case illustrates that OntoCAD supports the

incorporation of EVs not currently supported by CAD systems, where it focuses on the demonstration of modelling methodology; the second illustrates how an EV that is already incorporated in CAD systems may also be assisted by OntoCAD.

To be more explicit, the first experimental case is to integrate a commercial costing tool SEER-MFG with a CAD system NX6. The SEER-MFG tool needs parameter inputs to calculate cost results. The parameters are mainly two types: geometric data from CAD models (e.g. dimensions/volume, geometric features) and non-geometric data from user inputs or other software resources (e.g. quantity of production, material). In order to feed data to SEER-MFG, the FO as fundamental knowledge needs to be built, the EVO for cost needs to be built and integrated, so does the AO for SEER-MFG, and then populated with data either from users or software applications.

Considering a scenario illustrated in Figure 46, once the user initiates a cost analysis request, SEER-MFG will send a request to the OntoCAD system, in which the agent makes a judgement whether the current data is sufficient for SEER-MFG to operate. If not, the agent asks the user to provide more information. Meanwhile, the agent assigns an AW for this request to observe whether it can be satisfied. Once sufficient data become available, the OntoCAD agent will export the data to SEER-MFG to compute and then return a real-time result to the user. In an intended support, these processes are operated automatically as background processes, while in the actual support, user intervention is required.

In the case of FEA, the use case is very similar to the cost estimation case, but mainly demonstrates generality and efficiency of the ontology modelling methodology again in the case of integrating new AO and EVO into existing ontologies. In other words, it is more focused on the aspect of modelling knowledge of legacy tools rather than integrating another complex system into the OntoCAD paradigm. As introduced in Chapter 3, FEA tools and CAD system have been independently developed. A generic FEA works through three stages: pre-processing, solution and post-processing stage. The decision and input data are needed from the analyst to prepare the finite element model in the pre-processing stage, such as meshing instructions, boundary conditions, initial conditions, and loading. At the post-processing stage, interpreting of the analysis result may need to be recorded and referred to the CAD model. In this FEA case study, it demonstrates the OntoCAD system allowing customized knowledge expansion by integrating an EV into the existing MEV knowledge base. Based on this, it also demonstrates transferring a knowledge model that is tightly embedded in either a CAD system or FEA system into an independent knowledge base.

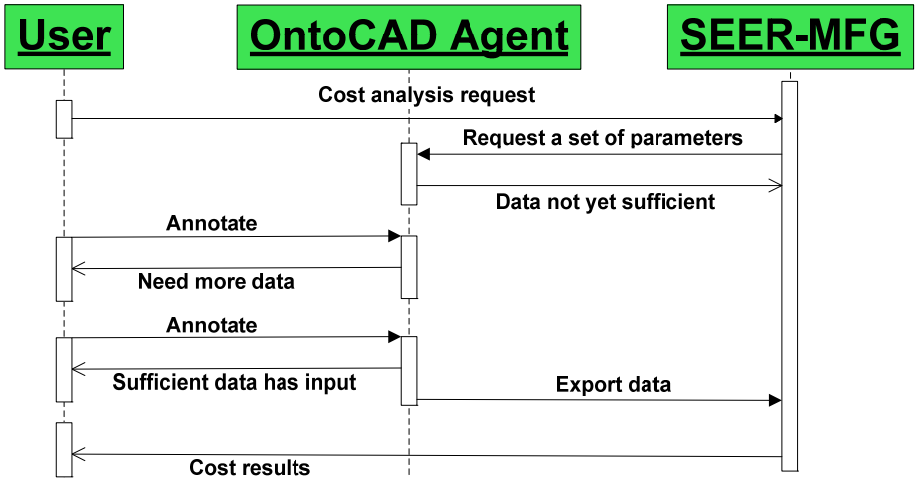


Figure 46 Sequence Diagram for the Integration of NX and SEER-MFG

According to the requirement list, the prototype system can be designed and broken down into three levels of task as illustrated in Figure 47. The task WBS 1.0 refers to the level-1 task in the work breakdown structure (WBS) – the overall experimental work, which is partitioned into three level-2 tasks for each OntoCAD system modules, and then the overall prototype system will be integrated and tested in a level-3 task – integration. Each task will be described in more detail in the following sub-sections.

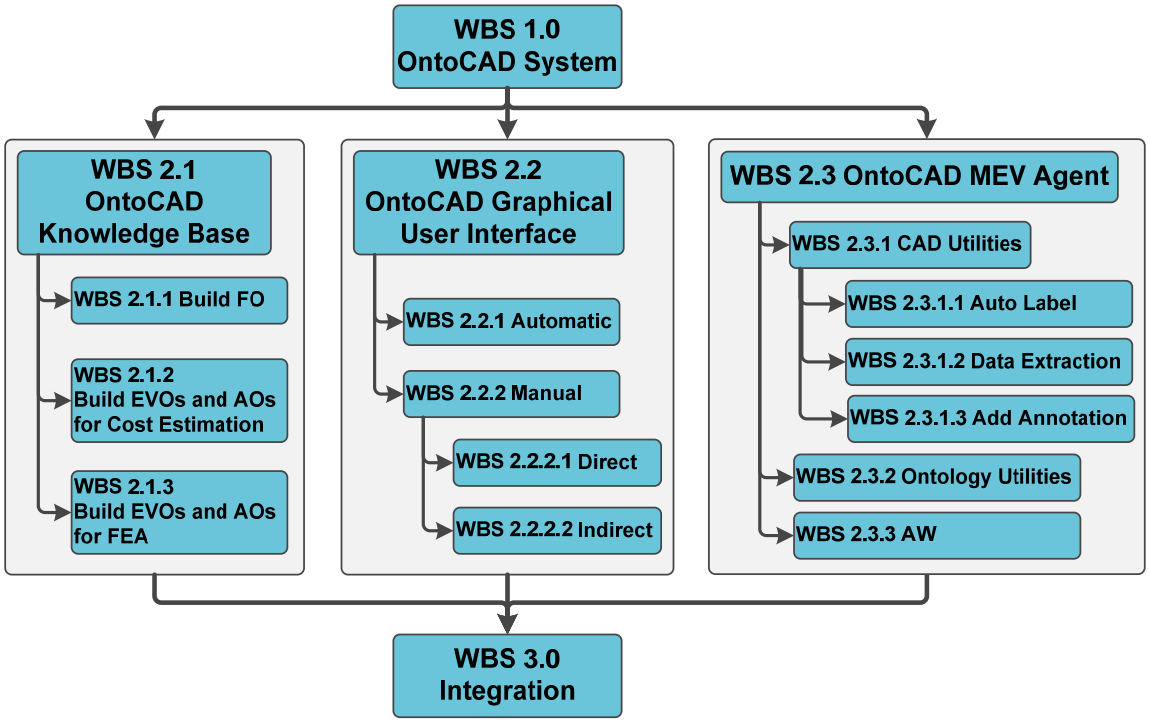


Figure 47 OntoCAD Prototype System Work Breakdown Structure

6.2.1 Design of the OntoCAD Knowledge Base

As the entire solution is driven by the knowledge base which is constructed by ontologies,

the design is started with the most fundamental module – the OKB. The task to build this knowledge base is mainly divided into three sub-tasks: build the common FO, and build EVOs and AOs for each case (Figure 47). For particular ontologies and their constituents, please refer to Section 6.3.

In an ideal situation, all FO, EVO, AO and their instances should be able to be saved independently and merged together only as needed. However, to ease the implementation, while all ontologies can be modularized conceptually they are saved as a whole in the experimental work.

6.2.2 Design of the OntoCAD Graphical User Interface

Considering the requirements for the OGUI in Table 32, the GUI should be able to capture annotation data in two ways: automatically and manually. In the automatic way (WBS 2.2.1 shown in Figure 47), all three types of geometric elements in a CAD model will be visited and automatically labelled as a preparation for anchors, so that the generated labels can be used as identifications for anchors. Another automatic process is data extraction from CAD model, in which a set of utilities in the OMA needs to access CAD related information. These automatic processes can demonstrate the ability of automatic knowledge acquisition from an existing CAD model and the granularities of manipulating geometric elements, which are essentially supported by the OGUI.

On the other hand, an interface that supports manual annotation will be developed in WBS 2.2.2 (Figure 47), which has two sub-tasks: direct annotation and indirect annotation. As noted in Figure 33 of Chapter 5, direct annotation implies the annotations directly associated with the most primary target – CAD models, while the indirect annotations are further chained annotations indirectly associated with geometries. The OGUI will handle the interface configuration instructions – ICIs, therefore to demonstrate the dynamic features that can adapt to context changes.

Theoretically an interface can be developed for defining and modifying AW rules, and assigning an AW, in order to bridge between the ontology editor Protégé and NX. However, considering saving time on experimental work, this feature is omitted, and will be demonstrated with separated preparation using Protégé. With regard to interfacing the AWs, it will be supported by OGUI.

6.2.3 Design of the OntoCAD MEV Agent

Different from the OGUI, the OMA is an agent – a means of interfaces and functions that

allows OntoCAD modules to interact programmatically. This implies facilities are needed, including satisfying queries to the knowledge base, and reasoning processes over the knowledge base as required. To achieve this, two sub-tasks WBS 2.3.1 and WBS 2.3.2 are allocated as shown in Figure 47. The first task – the CAD utilities provide access to CAD models, and the latter task – the ontology utilities provide access to the knowledge base.

Furthermore, the WBS 2.3.3 develops an agent to operate AWs through the utilities developed in WBS 2.3.1 and WBS 2.3.2. To operate an AW includes monitoring its status and invoking downstream processes, such as data exchange though the OGUI.

6.3 Implementation

Having specified the requirements for the actual support and designed the OntoCAD demonstration system, this section will describe the implementation in the actual experimental work. This mainly has four parts, the implementation for the OKB, OGUI, OMA and a final integration.

6.3.1 Implementation of the OntoCAD Knowledge Base

As the WBS in Figure 48 implies the OKB had three sub-tasks. The FO was implemented to construct a foundation for geometric models for the preparation of anchors and some other fundamental classes. And the other two sub-tasks were specifically built for the two case studies.

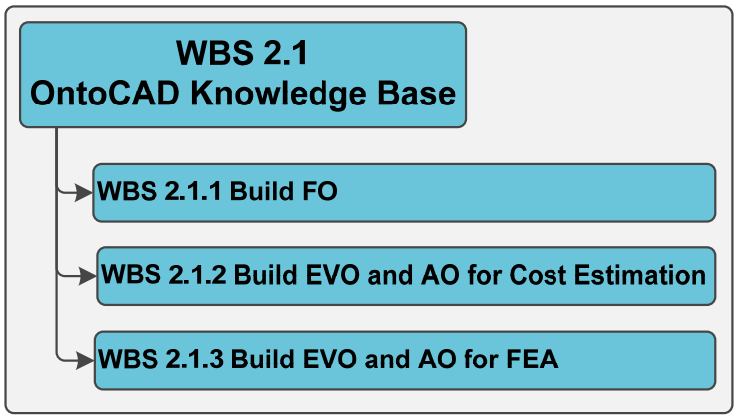


Figure 48 Work Breakdown Structure Sub-tasks for the OntoCAD Knowledge Base

Build FO

In order to address the direct annotation structure, associations were required between the geometric elements and ontologies, which have been introduced in the OntoCAD

annotation data structure (Chapter 5). Thus the most primary goal in the FO was to build a consensual knowledge on the anchoring mechanism, and the proposed solution was to use OWL as a formal Interlingua to describe geometric models and their constituents, and to comply with the STEP standard.

Since the FO was built from scratch, a middle-out modelling strategy was adopted. The guidelines described in Chapter 5 were followed, throughout knowledge acquisition, specification, conceptualization, integration, implementation, and evaluation; however the detailed phases will only be briefly described.

In the phase of knowledge acquisition, brainstorming meetings were held to initially identify the concerned fields, including the geometric boundary representation and measurement units. Following the literature research, the STEP parts concerned are listed in Table 33.

Table 33 Concerned Parts of the STEP Standard Family in the Present Work

Document	Source	Description
ISO 10303-1:1994	(ISO 1994a)	Part 1: Overview and fundamental principles.
ISO 10303-11:1994	(ISO 1994b)	Part 11: Description methods: The EXPRESS language reference manual. Definitions for data types mainly refer to this document. And it helps to understand how product data is specified.
ISO 10303-21:1994	(ISO 1994c)	Part 21: Implementation methods: Clear text encoding of the exchange structure. Definitions for data types mainly refer to this document.
ISO 10303-203:1994	(ISO 1994f)	Part 203: Application protocol: Configuration controlled 3D designs of mechanical parts and assemblies. B-rep modelling mainly refers to this document.

Based on these standards, some sub-ontologies were defined through specification, conceptualization, integration, implementation, and evaluation as depicted in Figure 49, where some details are omitted for the reason of conciseness, and can be reviewed in the ontology metadata and other project documents[10]. The blocks in blue denote FO classes. For example, “shape_representation” is a class that forms one of the definitions “*hasShapeRepresentation some shape_representation*” for the class “Part”, where the cardinality between part and “shape_representation” is one to many (i.e. at least one). The blocks in light blue denote the sub classes of an associated super-class (linked with

[10] The digital ontology metadata and the OntoCAD prototype software application are available online from http://dl.dropbox.com/u/45907729/OntoCAD.rar (Li 2012).

dotted lines). The round blocks denote individuals of OWL classes, associated through "is_a" properties.

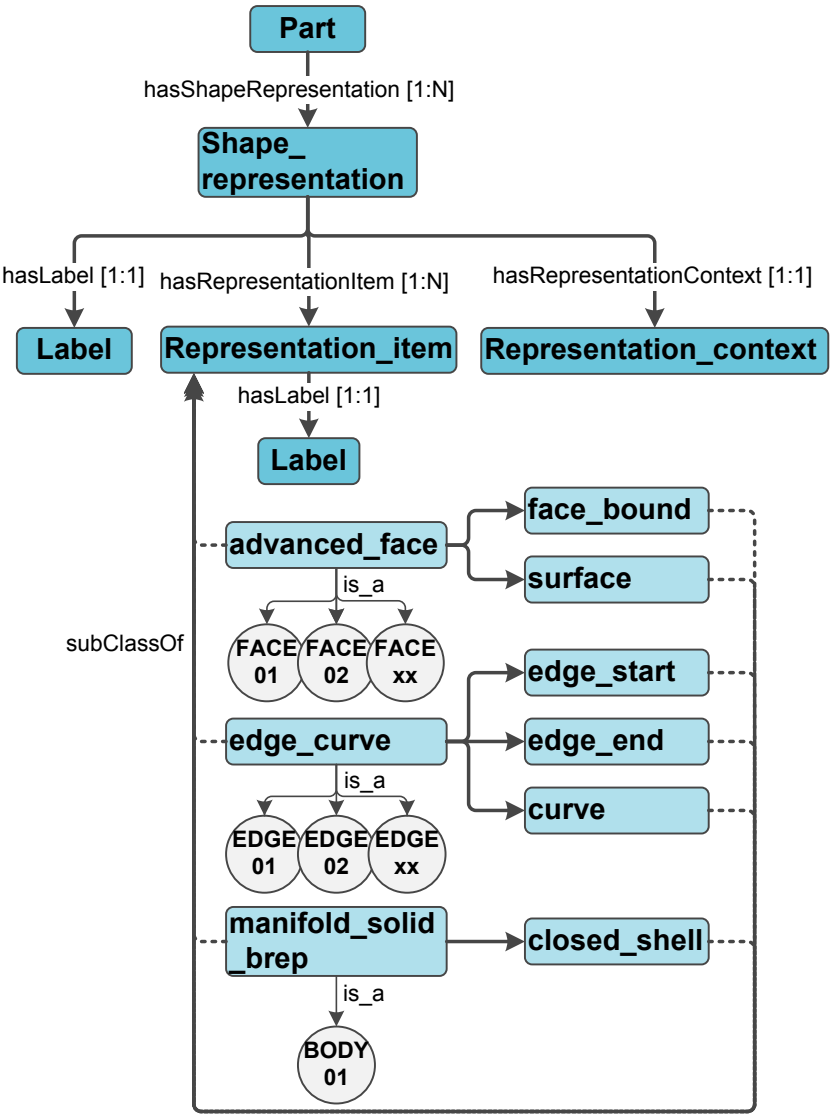


Figure 49 Partial View of the FO for OntoCAD Annotation Anchors

Apart from the anchor related FOs, there are also other fundamental classes. Although some of them are patched during the development of other EVOs and AOs, they are still described here as they compose of the FO. Two notable FO classes are "measure_with_unit" and "data". As Figure 50 illustrates, a measurement is generally composed of a measure value (or values) and a corresponding unit (or units), complied with STEP standards. Data has many different types, including partially listed: binary, integer, string, real and so on. "measure_value" may associate with a single data type – real number. On the other hand, a unit may have a sub type – a named unit that has many other sub types: si_unit, length_unit, area_unit and others. "si_unit" has two attributes: "si_unit_name" and "si_prefix". And all sub classes of "named_unit" inherit a common

attribute “dimensional_exponents”, which has attributes to define derived units other than “si_unit”: length_exponent, mass_exponent, time_exponent, electric_current_exponent, thermodynamic_temperature_exponent, amount_of_substance_exponent, and luminous_intensity_exponent.

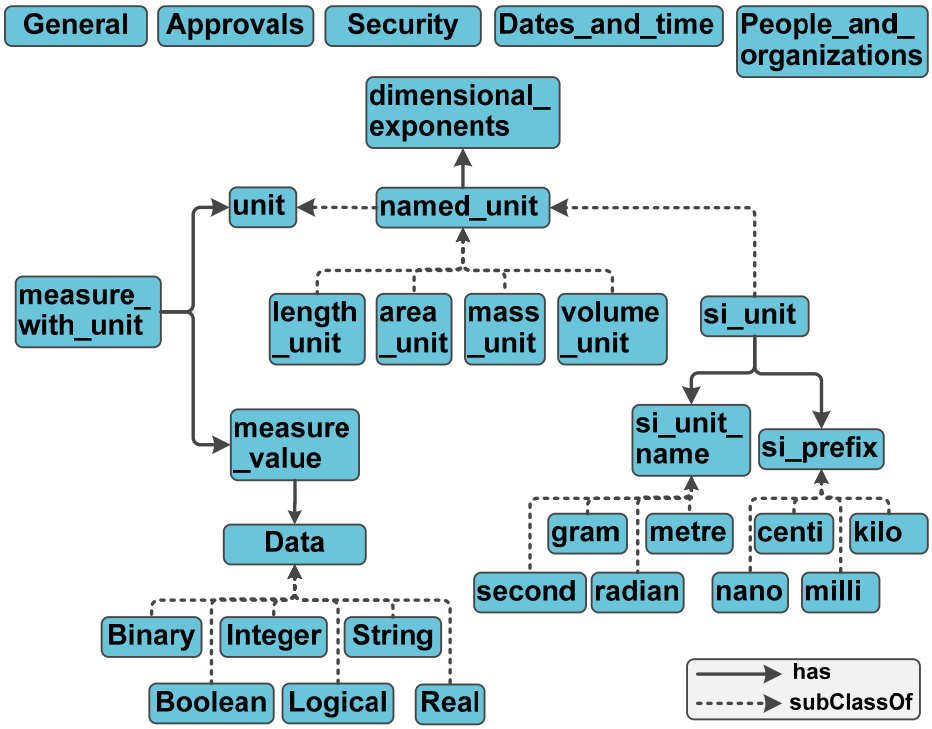


Figure 50 Partial View of the FO for Non-Geometric Classes

There are also other FO classes, such as “General” for general information of a design model, “Security” relating to issues for access control, and so on.

Build EVOs and AOs

The corresponding ontologies representing EV knowledge of cost estimation and FEA are implemented, and two different scenarios were taken for the purpose of evaluating knowledge modelling methodology.

Scenario 1 (WBS 2.1.2):

In this scenario, EVO_Cost for cost estimation does not exist and is built independently for general costing analysis. And the AO_SEER for the SEER-MFG tool is then built and mapped to EVO_Cost.

The first phase in this scenario was to model EVO_Cost, where the middle-out strategy was adopted. The EVO_Cost ontology aims to describe a set of effective cost drivers in

general, rather than exploring and defining algorithms for estimating costs. The Delphi method (Skulmoski et al. 2007) was used to identify cost drivers in order to scope and conceptualize this EV in the early stages of knowledge modelling.

Based on a literature review, a first round questionnaire was designed with a list of candidate cost drivers that may potentially affect manufacturing cost and this was distributed to a group of academic cost experts. For example, a list of candidates for the sand casting manufacturing process includes production quantity, direct labour hour rate, material, finished weight, tool description, inspection/rework, and so on.

Based on the feedback from the first round questionnaire, some common cost drivers were identified. Having validated these against the commercial cost modelling tool SEER-MFG, a second round questionnaire was produced in order to ask interviewees to identify the most significant cost drivers to concentrate on in the experimental work. In the second round feedback, cost drivers confirmed as the most significant factors included production quantity, material, finished weight, etc. in the case of sand casting. Through this Delphi method, a set of cost drivers and basic cost rules were modelled as an EVO and incorporated into the OKB. However, this ontology only covers a selection of manufacturing processes since completeness of this EVO_Cost is not the primary goal.

In the cost ontology, as shown in Figure 51, cost has two attributes in general – the value and unit. The types of costs are classified into three main categories: labour, material and tooling costs. Since engineering viewpoints can not stand alone, and each is affected by a number of other classes or ontologies, some other EVOs were also coarsely defined for the purpose of demonstrating EVO_Cost only. For example, manufacturing processes have an affect on labour cost, as well as tooling cost, while material selection, part weight and shape representation affect material cost. For reasons of conciseness, not all ontology classes and interconnections are depicted in this diagram, neither are detailed relations between classes and subclasses. For instance, there is a relationship between material density (material property) and part dimensions (shape representation), from which weight can be computed in order to evaluate material cost. Furthermore, the direct annotation associativity and granularity constraints (G1, G2 and G3) are defined. G1 indicates that this class can associate with the highest level of geometric representation, namely a body, while G2 comprises face(s) and G3 elements include edge(s). The illustrated classes in two different blocks are to differentiate the standard-compliant and non-standard-compliant classes.

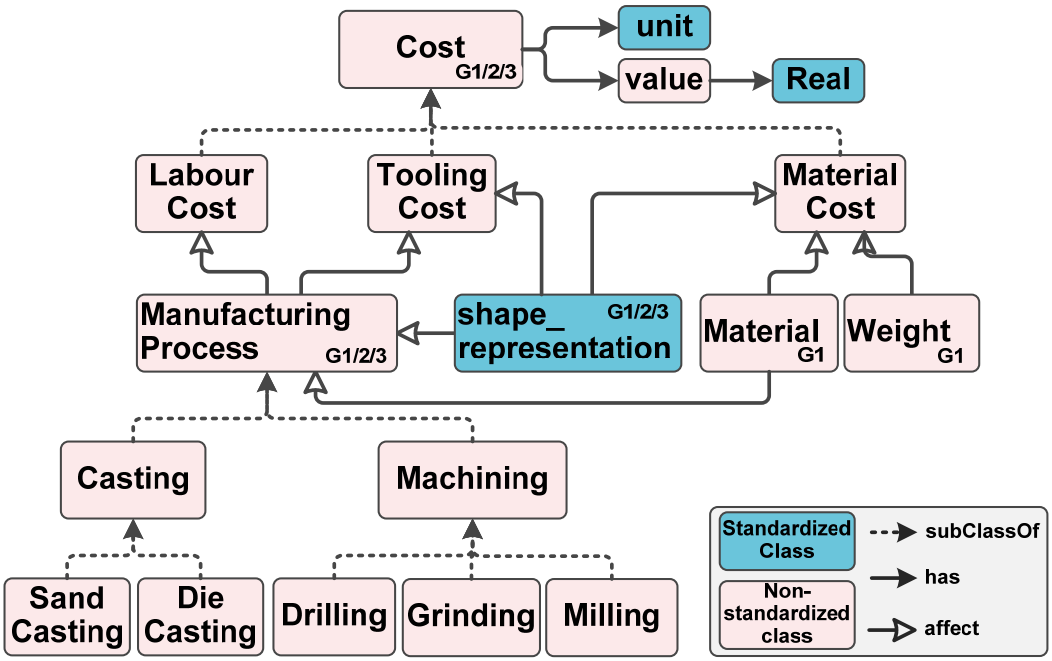


Figure 51 Partial View of Cost EV Ontology

As noted, the actual task of cost estimation is achieved by retrieving data from the OKB and passing the annotation data to the external engineering tool SEER-MFG to compute the cost results in terms of labour cost, material cost and other additional costs. In the experiment, the set of the most significant cost drivers was applied to a passenger vehicle part design, which is a towbar manufactured using die casting, drilling, grinding and paint spraying as a finishing treatment. The required data can be retrieved and passed to SEER-MFG by applying mappings between the FO, AO and EVO. In order to achieve the semantic retrieval, the application ontology AO_SEER was modelled in the second phase in the Scenario 1, where the bottom-up strategy was adopted.

- The specification of the application can be produced by developing the requirement specification document, GT document, and documents for table of classes, properties, axioms and rules as introduced in Chapter 5 (as illustrated in Table 22, Table 23, Table 24, Table 25, Table 26, Table 27 and Table 29).
- The gathered terms are then generalized, mapped and updated to existing EVOs and FO by following the guidelines introduced in Chapter 5 (Table 30). New EVOs or components of FO can be patched if necessary.

- The AO_SEER is then coded using Protégé, evaluated (verification only yet) and documented.

The resulting AO_SEER is partially illustrated in Table 34, showing the mappings appropriate to the example part.

Table 34 Definition Mapping for the SEER Application Ontology

Ontology Level	Corresponding Class	Definition in AO_SEER
EVO	EVO_ManufacturingProcess	PRODUCT_DESCRIPTION_-_Process
EVO	EVO_Material	PRODUCT_DESCRIPTION_-_Material
FO	Weight	PRODUCT_DESCRIPTION_-_Finished_Weight
FO	Quantity	ProductionQuantity
EVO	SandCasting (EVO_ManufacturingProcess)	Sand_Casting
EVO	DuctileCastIron (EVO_ManufacturingProcess)	Ductile_Cast_Irons Iron_Cast_,_Ductile
EVO	DirectHourlyLabourRate(EVO_Cost)	PRODUCT_DESCRIPTION_-_Direct_Hourly_Labor_Rate

Scenario 2 (WBS 2.1.3):

As noted in Chapter 3, a generic FEA works through three stages: pre-processing, solution and post-processing stage and decision and input data are needed from the analyst to prepare the finite element model in the pre-processing stage, such as meshing instructions, boundary conditions, initial conditions, and loading. All this information may be associated with a geometric model as annotation. FEA tools can be tightly coupled with CAD systems, either embedding at least an FEA pre-processor into CAD system or a geometric modeller into FEA tool, in which the first case is chosen in this present work. The use case is to transform knowledge of a commercial FEA tool ANSYS® into the OntoCAD system so that the necessary input for FEA pre-processing and interpretation of the output from FEA post-processing can be stored as integrated generic knowledge. This gives this generic FEA knowledge model the potentials to be used by other EVs or transferred from one specific FEA tool to another.

The main purpose of this case study is to verify the efficiency of the ontology modelling methodology again in the case of integrating new AO and EVO into the OKB, and to demonstrate knowledge of legacy engineering tools that are already coupled with CAD systems can be incorporated, rather than integrating another complex system into the OntoCAD paradigm. In this scenario, an EVO for FEA does not exist, and neither an AO for FEA tool. Considering the sophistication of current commercial FEA tools, it is not

necessary to build an EVO for FEA beforehand. Instead, one of the leading FEA tools can be used as a starting point to establish the AO, which can then be generalized to a corresponding EVO. Therefore, a bottom-up strategy is taken to build an AO for ANSYS® (AO_ANSYS) and an EVO for FEA (EVO_FEA). Furthermore, FEA dominates many application areas, including stress, thermal, fluid flow and others as noted before. Due to this comprehensiveness, only partial aspects of structural stress and thermal analysis were implemented in this case study.

The AO_ANSYS was modelled in a similar way to the AO_SEER by extracting terms from the manual book and the application itself. The EVO_FEA was then generalized from AO_ANSYS classes. Figure 52 illustrates a schematic partial view of example EVO_FEA, AO_ANSYS and the synergy among all levels of ontologies, in which the upper blocks represent EVO and its constituents with properties, the blocks on the right represent FO, and the lower blocks represent AO_ANSYS with the terms specifically defined for ANSYS® tool, e.g. “ConstantValue” is mapped as an equivalent to data type “Real”.

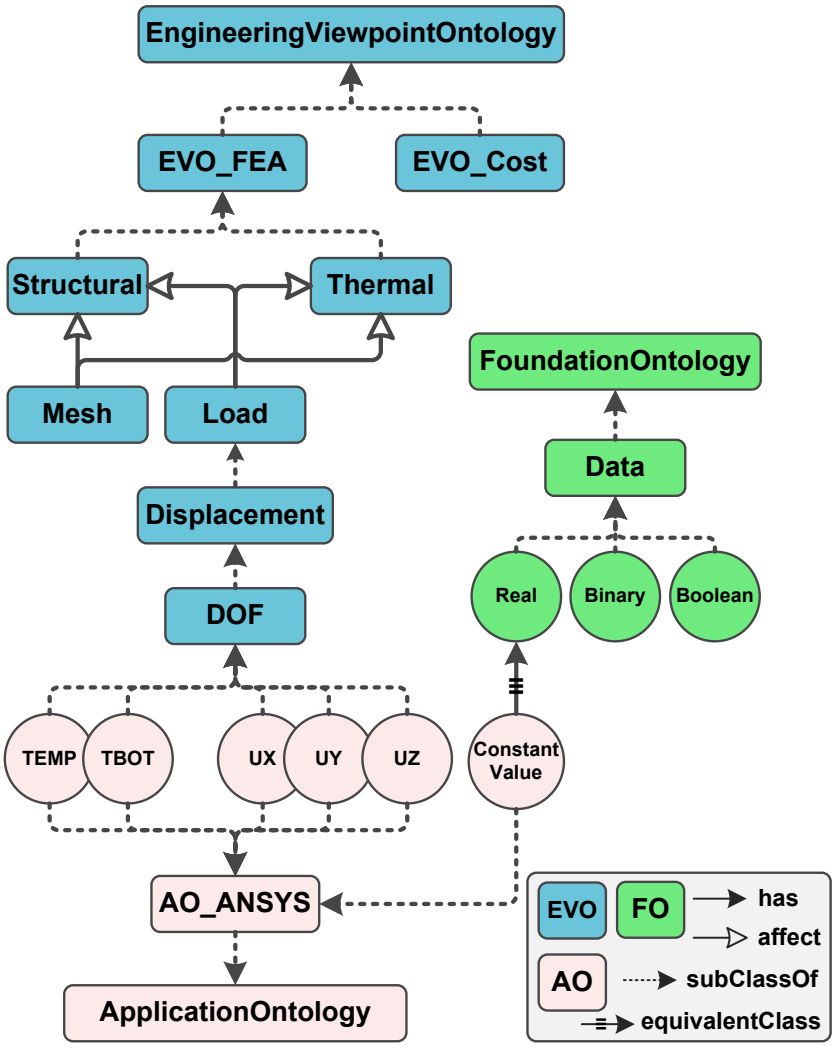


Figure 52 Partial View of the Collaboration among EVO and AO for FEA and FO

In the case of cost estimation, the OMA can explicitly understand the queries made for SEER, which are satisfied with accurate semantic data retrieval and then fed back to SEER with the help of methodological reasoning. In the case of FEA, the actual implementation did not try to couple ANSYS® with NX, but to test information can be annotated on CAD models and recorded in the OntoCAD model, such as initial conditions, loading etc. For example, the “ux”, “uy” and “uz” for defining “DOF” (degrees of freedom) can be annotated and stored.

Table 35 Overview of the OKB MEV Ontologies and Exemplary Classes

Ontologies (# of classes)	Primary Classes (# of Subclasses)	Exemplary Subclasses (# of Subclasses)
AW (1)	AW_SEER_1 (0) AW_SandCasting_Rule_1 (0)	
AO (2)	AO_SEER (4)	PRODUCT_DESCRIPTION_-_Finished_Weight_(kg) (0); PRODUCT_DESCRIPTION_-_Process (2)
	AO_ANSYS (6)	ConstantValue (0); DOF (12) ElementType (0); ElementTypeOption (3); Loads (5); Meshing (1)
EVO (5)	EVO_Cost (2)	Cost (3); CostDriver (0)
	EVO_FEA (2)	Structural (0); Thermal (0)
	EVO_Manufacturing (1)	ManufacturingProcess (7)
	EVO_Material (2)	Material (7); MaterialProperties (5)
	EVO_Design (1)	Note (0)
FO (12)	Comment (0)	
	Data (8)	Boolean (0); Integer (0); Real (0); String (0); EnumerationValue (0)
	Part (0)	
	dimensional_exponents (0)	
	measure_value (12)	area_measure (0); mass_measure (0); count_measure (0)
	measure_with_unit (1)	Weight (0)
	representation_context (0)	
	representation_item (103)	advanced_face (0); edge_curve (0); manifold_solid_brep (0)
	shape_representation (7)	advanced_brep_shape_representation (0); edge_based_wireframe_shape_representation (0)
	si_prefix (16)	kilo (0); mega (0); micro (0); milli (0)
	si_unit_name (28)	hertz (0); metre (0); newton (0); watt (0)
	unit (1)	named_unit (5)

Summary

As Table 35 shows, this prototype OKB mainly contains the three layered MEV ontologies (i.e. FO, EVO and AO) and one extra layer of OMA application (i.e. AW), which consists of 20 primary classes so far. Each primary class may contain sub-classes with multiple levels. Although some classes indicate as no further subclasses, they may have inferred subclasses after conceptual reasoning actions according to their defined conditions. For example, the class CostDriver under EVO_Cost is defined as any class affect costs. As a result, it has eight inferred subclasses computed by the reasoner. To be noted, the naming convention implies that names in camelCase style refer to classes conceptualized by the authors, while names start with lowercase letters and/or with underscores are standard-compliant or tool-compliant. For further detail of the ontology metadata, please refer to the digital resources (Li 2012).

Both the OWL object and data properties were used in this prototype OKB. With regard to relation types, object properties can be classified into three types as shown in Table 36: “has”, “is”, and “affect”. The “is” category is a collection of inverse properties of “has”. The “has” category was mainly used to construct associations between individuals. The “affect” category was mainly used to construct conditions for equivalency of classes, on which based conceptual reasoning operations can be computed.

Data properties can be classified into two types as shown in Table 37: “has”, and “is”. The “has” category is a collection of data properties that construct association between an individual and data. The “is” category is not inverse properties, but flag properties (normally indicate the states of a parameter), for example, “isApplicationWatchdogReady” is a Boolean property that indicates the status of an AW. Each property may contain further sub-properties.

Table 36 Object Properties and their Sub-Properties

Property Categories	Properties	Sub-properties
affect	affectCost	
	affectFEA	affectStructural, affectThermal
has	hasCost	
	hasDOF	
	hasData	
	hasEdgeStart	
	hasEdgeEnd	
	hasElement	
	hasElementType	
	hasElementTypeOption	hasK3, hasK5, hasK6
	hasLoad	hasDisplacement, hasPressure
	hasManufacturingProcess	
	hasMaterial	
	hasMaterialProperty	
	hasUnit	
	hasUnitName	
	hasUnitPrefix	
	hasWeight	
is	isElementOf	
	isMaterialOf	

Table 37 Data Properties and their Sub-Properties

Property Categories	Properties	Sub-properties
has	hasComment	
	hasID	
	hasLabel	
	hasValue	hasRawMaterialCostValue
		hasElementDivisions
		hasElementEdgeLength
		hasOptionValue
		hasWeightValue
is	isApplicationWatchdogReady	

6.3.2 Implementation of the OntoCAD Graphical User Interface

As noted in Section 6.2.2, the OGUI needs to satisfy two mechanisms of knowledge acquisition: automatic and manual. As a consequence, the development of the OGUI is divided into two subtasks (Figure 53). The prototype OGUI was embedded in NX6 as an add-on GUI, which was developed with two types of interfaces – menu and dialog.

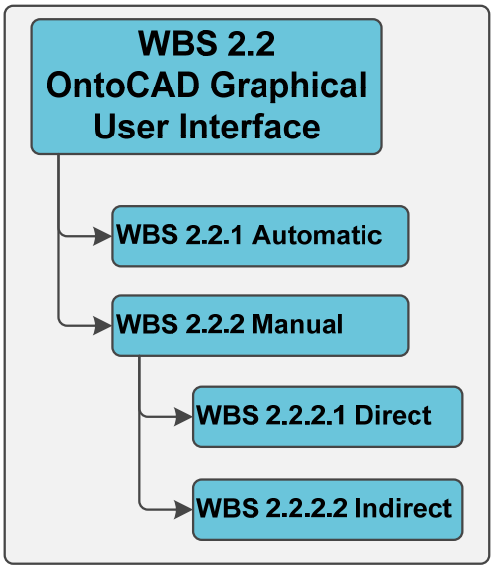


Figure 53 Work Breakdown Structure Sub-tasks for the OntoCAD Graphical User Interface

- *Menu*: Menu files of NX6 support custom tailoring of the main menu bar and the Quick View Popup menu. The menu was developed by using NX Open Menuscript provided by NX6 (Siemens PLM Software Inc 2008). It lets users use ASCII files[11] to edit NX menus, and to create custom menus for their own applications. Some key features include creating or modifying cascade/toggle buttons, positioning buttons, showing and hiding buttons, defining button actions and so on.
- *Dialog*: NX dialogs can be developed using the Open User Interface Styler and launched from a MenuScript menubar. The Open User Interface Styler is a visual dialog box builder for NX users to build dialogs. And customized tasks can be performed by executing "callback" functions upon user clicks on dialog items. Many programming languages are supported in NX Open for writing callback functions, such as C/C++ and JAVA. NX Open will be briefly introduced in Section 6.3.3.

The interaction among menus, dialogs, and callback functions is depicted in Figure 54. The customized menu can be amended to the main menus and be used to launch

[11] American National Standard Code for Information Interchange (ASCII) files refers to "plain text" computer files that are coded in a widely accepted character-encoding scheme – ASCII. (American National Standards Institute 1986)

customized dialogs, which can execute callback functions to perform user defined functions.

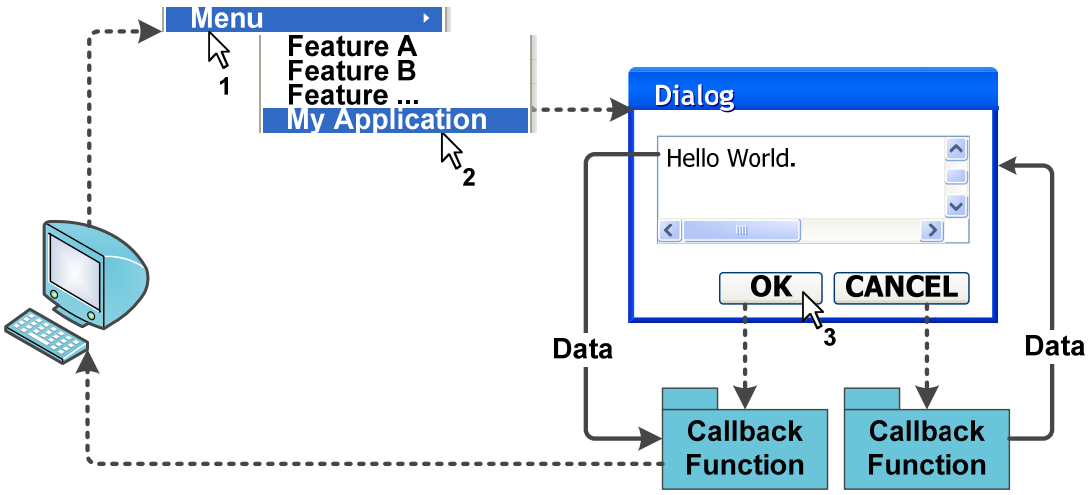


Figure 54 OntoCAD Graphical User Interface Interactions (Siemens PLM Software Inc 2008)

WBS 2.2.1 GUI for Automatic Knowledge Acquisition

The GUI for automatic knowledge acquisition has two modules: automatic labelling and annotating. The automatic data extraction from CAD models did not fall in this scope as it is a set of functions operated at background rather than a GUI, which will be described in WBS 2.3.1 CAD Utilities.

The first module is for automatically labelling all concerned geometric elements of a CAD model at three levels of granularity, so that the generated labels can be used as anchors. This was implemented using menus as shown in Figure 55. The button “Start” is an original NX GUI item, which leads to a menu launching the customer application, in this case the “OntoCAD”. The menu item “OntoCAD” (Figure 55 (a)) can enable OntoCAD in the main menu bar (Figure 55 (b)). The “OntoCAD” button has two drop-down menu items: “Auto Label” for automatic knowledge acquisition (programmatically walk through all geometric elements and label them) and “Add Annotation” for manual knowledge acquisition (to add user annotations).

The second module is an interface to accommodate AWs which check whether a query or engineering rule enters a satisfied state, thus a dataset can be compiled and made ready for exchange. The “Start AW” button (Figure 55 (b)) is a drop-down menu item of “OntoCAD-AW”, which is responsible to enable the AW. Once the OMA notifies an AW is satisfied, a dialog can be invoked by a user in order to input a spreadsheet containing queries thus to populate a dataset into the spreadsheet for downstream applications to

process.

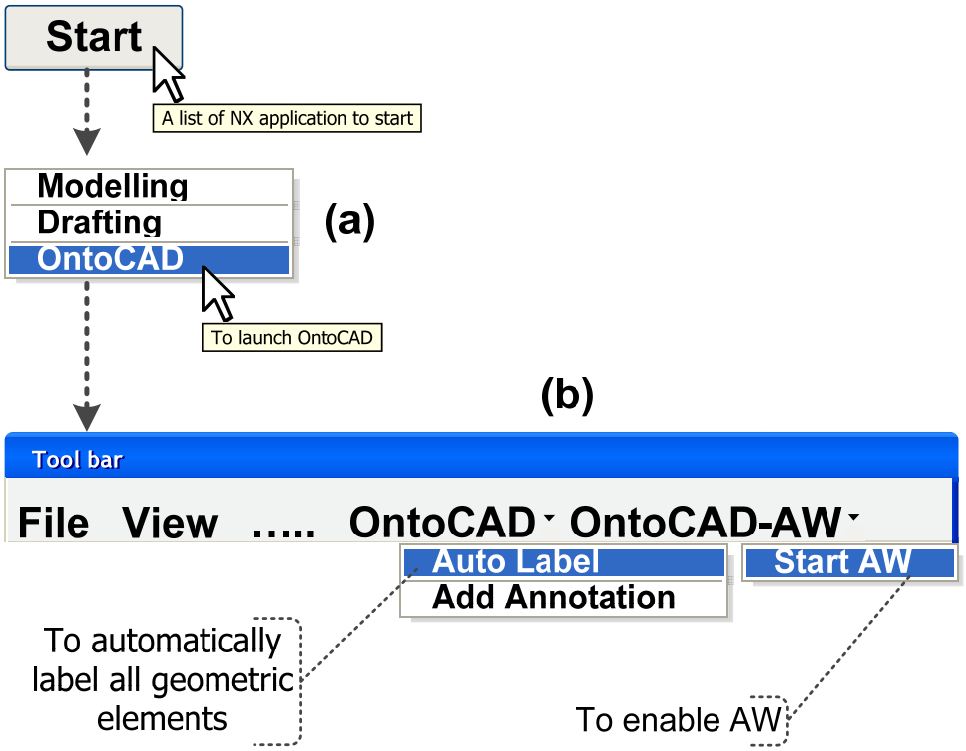


Figure 55 OntoCAD OGUI Menus

In the actual implementation, Figure 56 shows the example Menuscript file for a menu illustrated in Figure 55 (a), in which the keyword "MENU_FILES" links to another Menuscript file (Figure 57) for button "OntoCAD" in Figure 55 (b). In the second Menuscript file shown in Figure 57, the callback function "OntoCADAutoLabelCB" invokes the automatic labelling task, which will be described in Section 6.3.3.

```
VERSION 120
EDIT UG_GATEWAY_MAIN_MENUBAR
MENU UG_APPLICATION
    APPLICATION_BUTTON ONTOCAD
    LABEL OntoCAD
    LIBRARIES OntoCAD.jar
    MENU_FILES ONTOCADANNOTATIONS_APP.men
END_OF_MENU
```

Figure 56 Example Menuscript for Main Menu Bar

```
VERSION 120
EDIT UG_GATEWAY_MAIN_MENUBAR
    ! *** position the button "OntoCAD" ***
     BEFORE UG_HELP
     CASCADE_BUTTON UISTYLER_DLG_CASCADE_BTN
     LABEL OntoCAD
     END_OF_BEFORE

     MENU UISTYLER_DLG_CASCADE_BTN
! *** create button "Auto Label" and link to callback function
     TOGGLE_BUTTON AUTO_LABEL_BUTTON
     LABEL Auto Label
     ACTIONS OntoCADAutoLabelCB

! *** create button "Add annotation" and link to a java library package
     BUTTON ONTOCADADDANNOTATION_BTN
     LABEL Add annotation
     ACTIONS OntoCADAddAnnotation.jar
END_OF_MENU
```

Figure 57 Example Menuscript for a Drop-Down Menu

WBS 2.2.2 GUI for Manual Knowledge Acquisition

As noted, the button "Add Annotation" in (Figure 55 (b)) is an access to manual knowledge acquisition. Upon the click of this button, dialogs can be launched for users to add direct annotations or indirect annotations.

WBS 2.2.2.1 GUI for Direct Manual Knowledge Acquisition

As noted before, the direct manual knowledge acquisition refers to allowing the user to directly associate annotations with the geometric elements. Therefore, clicking on the button "Add Annotation" in (Figure 55 (b)), will lead user to the dialog shown in the screen shot – Figure 58. The direct dialog has three tabs: anchor, engineering viewpoint, and annotation data. The anchor tab allows user to interactively select a geometric element, where the button "select" locks user selection, and the button "change" enables user to unlock one's selection and make changes. The engineering viewpoint tab allows user to orientate their EV by selecting from an available list, which is driven by the OKB. The annotation data tab is then affected by the previous EV selection, and changes accordingly, so that allow user to enter appropriate annotation content.

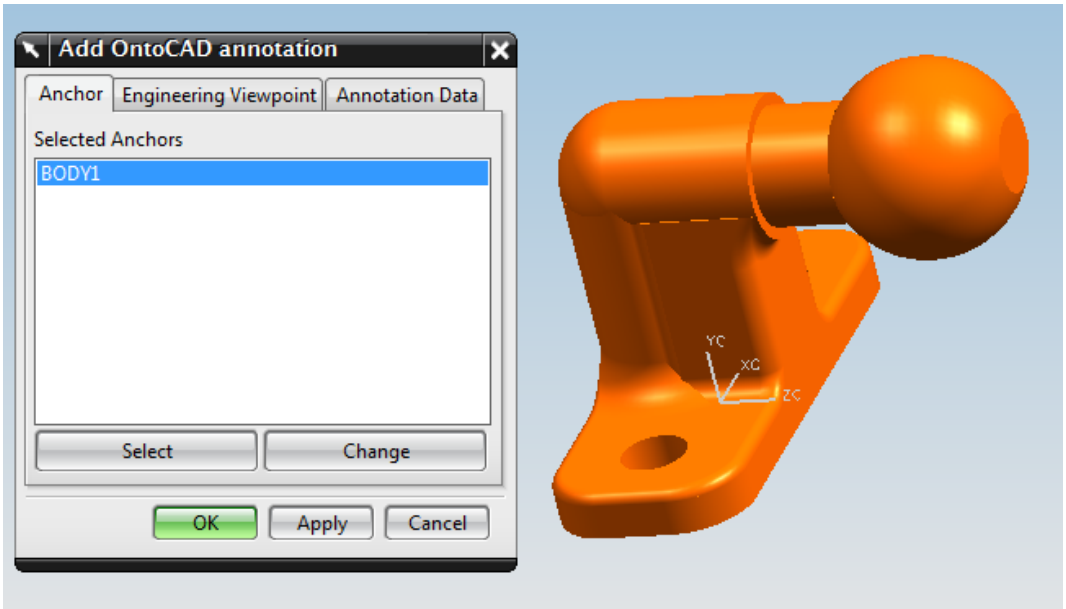


Figure 58 OntoCAD Dialog for User to Select an Anchor

This dialog and also other dialogs were built by using the Open User Interface Styler and can be launched through following steps:

1) Design a dialog on paper.

2) Build a dialog with the feature of drag-and-drop components from the GUI library.

3) Associate a dialog Item with a callback function.

4) Save the dialog.

5) Copy the Open User Interface Styler file and callback package to appropriate add-on working directory.

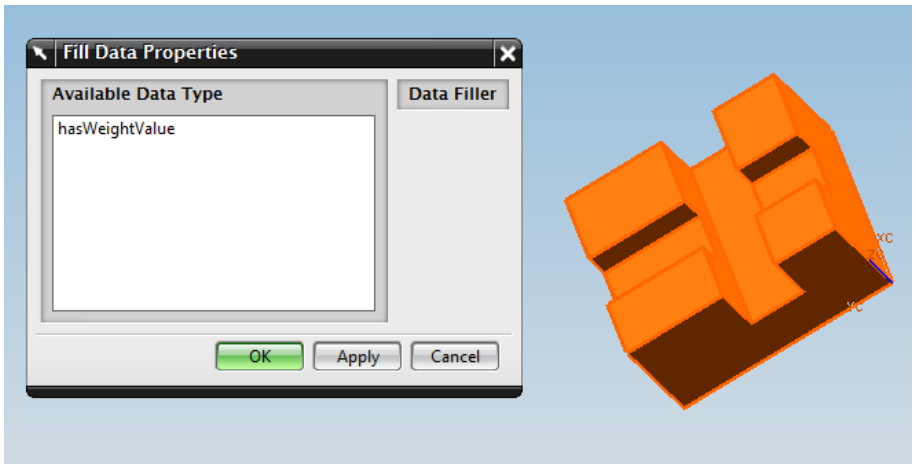
6) Launching a dialog from the menu bar (or launching a dialog from a callback or another dialog).

7) Execute, test, and debug.

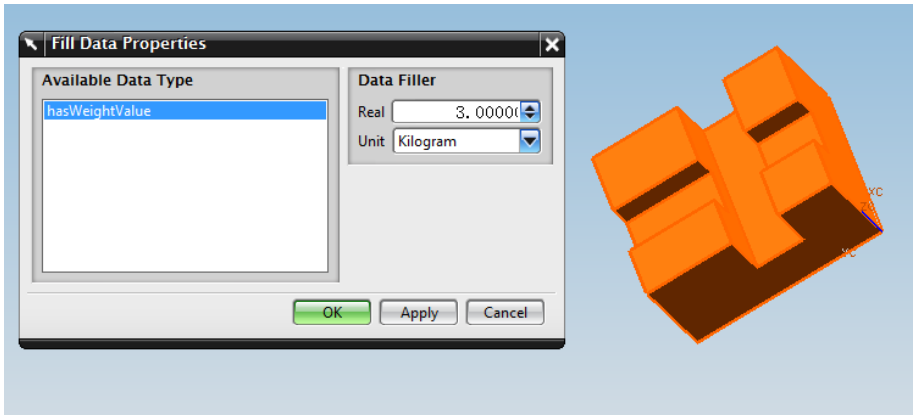
WBS 2.2.2.2 GUI for Indirect Manual Knowledge Acquisition

This WBS task emphasises the ability to assist with indirect annotation and the dynamic control of the GUI. A second dialog was developed as depicted in Figure 59. The screenshot (a) in Figure 59 illustrates a dialog in idle state launched from the dialog shown in Figure 58 when a data value is to be input by a user. The screenshot (b) illustrates the interface has changed accordingly when a specific data property type is

selected, and the appropriate data value type and corresponding unit are enabled.



(a)



(b)

Figure 59 Dynamic GUI for Filling Annotation Data

6.3.3 Implementation of the OntoCAD MEV Agent

As shown in Figure 60, the OMA deals with three aspects within the OntoCAD system depicted in Figure 27: CAD utilities (upstream), ontology utilities (interact with OKB horizontally), and AW to exchange with external tools (downstream). The software application is programmed in the JAVA language, with the employment of APIs, including NX Open for access to the CAD system (Siemens PLM Software Inc 2008), the OWL API for access to the OWL ontologies (University of Manchester et al. 2011), the Pellet reasoner to provide reasoning service for OWL ontologies (Clark & Parsia 2011), and a Java API for access to Excel spreadsheets (Khan 2012). Where, why and how these APIs have been used will be described in this section.

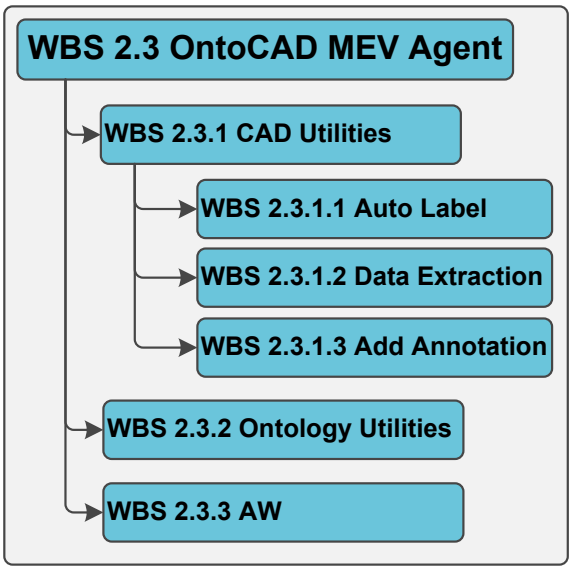


Figure 60 Work Breakdown Structure Sub-tasks for OntoCAD MEV Agent

WBS 2.3.1 CAD Utilities

The CAD utilities mainly include a mechanism that retrieves knowledge/information from the CAD system. According to Iyer et al. (2006), information embedded in CAD models can be classified into three categories: syntax (e.g. geometry, dimension, tolerance, feature), semantics (e.g. function, objective, constraint, manufacturing, maintenance), and pragmatic (e.g. corporation type, designer information, project information). All this information can be retrieved from CAD models automatically or manually. In conjunction with ontology utilities, the extracted information can be saved as instances into ontologies, which complete the process of data population.

In this experimental work, automatic labelling of all geometric entities at three levels of granularity was used as a case to demonstrate retrieval of geometric entities and preparation of anchors. In addition, a set of utilities to automatically extract information from CAD models was also developed such as weight, volume and area. This was a case to demonstrate that a CAD system and the OntoCAD system can be coupled more seamlessly through automated data population in the OKB module (i.e. knowledge integration). Adding annotation data specified in OWL to the OKB was also used as a case to demonstrate manually expanding the knowledge base through the CAD system interface.

WBS 2.3.1.1 Auto Label

Figure 61, illustrates how an automatic labelling process is performed in the callback function “OntoCADAutoLabel” once the menu item “Auto Label” is clicked (Figure 55 (b)).

This function starts with opening a CAD model and initializing the ontologies in the OKB. It then processes the three types of geometric elements, finalises the ontology (populates instances) and saves the CAD model with new labels. In the states of processing faces, pseudo codes in Figure 62 describe a process in performing the task, which can be seen as a general approach to process bodies, faces and edges. It should be noted that all OKB related tasks are performed by the ontology utilities and will be described in WBS 2.3.2.

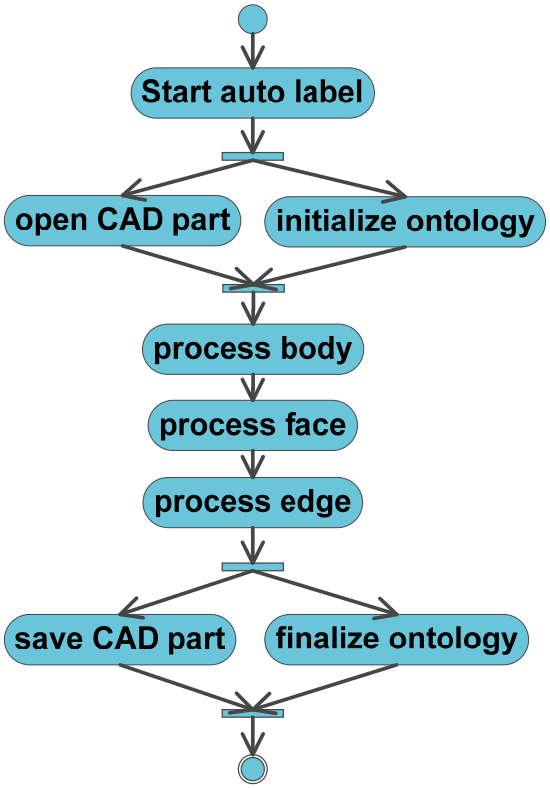


Figure 61 UML Activity Diagram for CAD Utility – Auto Label

```
for (all face)
{
    Create a label for the current face as ("FACE_"+ID);
    Add the label as an attribute to the current face;

    Create an individual of OWL class "advanced_face"
    Add this individual "FACE_XX" to the ontology;

    Get next face;
}
```

Figure 62 Pseudo Codes for Labelling All Faces in a CAD Model

WBS 2.3.1.2 Data Extraction

Other than preparation of geometric anchors, other information associated with CAD

models such as information of volume, mass or area for solid bodies or faces can also be extracted and populated into the OKB in order to improve the level of automation. This automatic process can be pre-population for efficient data collection, but may neglect up-to-date changes in geometric models. OntoCAD took a combination of these two strategies as illustrated in Figure 63, namely the pre-population process is performed at first, and then the OKB is updated again in real-time if modification of the CAD model is detected. Whether or not to perform a real-time process depends on whether the timestamps of the most recently modified version of a CAD model and the corresponding OKB are identical or not. The main flows in both strategies are similar to the automatic labelling process. The difference is executing an exemplary set of data extraction functions (Table 38) rather than naming geometric entities.

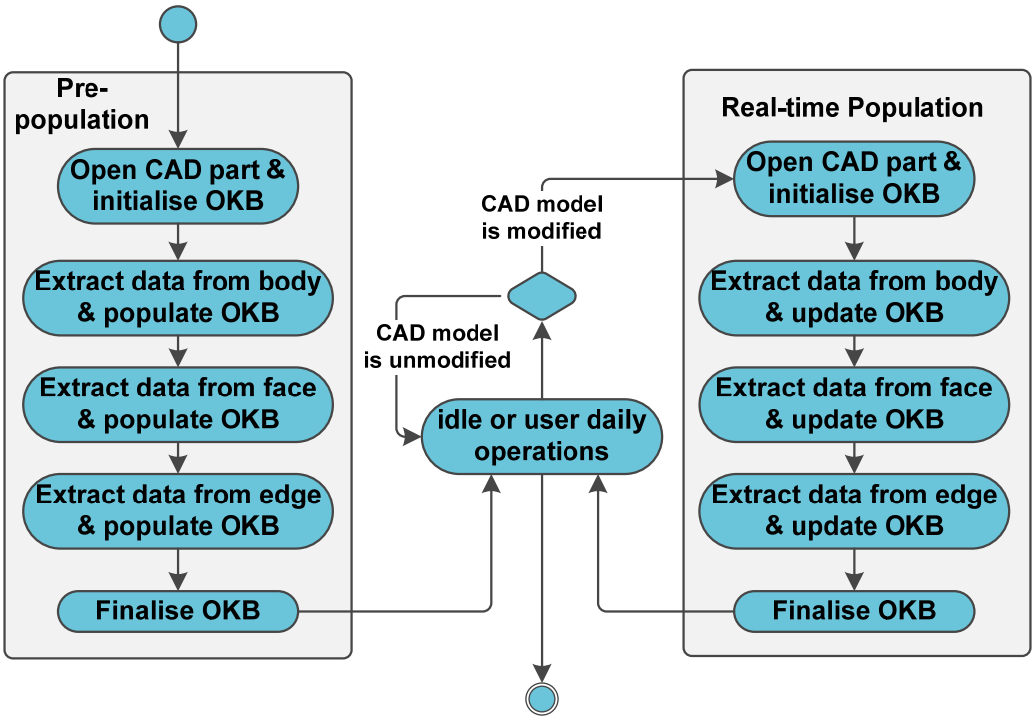


Figure 63 Automatic Data Population from CAD model into the OKB

Table 38 An Exemplary Set of Data Extraction Functions for CAD Models

Granularity	NX6 OPEN API Functions
Body	area (); mass (); volume (); weight ();
Face	area (); perimeter ();
Edge	length ();

WBS 2.3.1.3 Add Annotation

In the WBS 2.3.1.3, the UML state chart in Figure 64 illustrates how the callback function "OntoCADAddAnnotation" is executed when the menu item "Add annotation" is clicked (Figure 55 (b)). In the first state, a user can select or change anchor(s) since all anchors have been assigned with labels previously. Once anchors are selected, the user can select or change an EV from the available list. This list is dynamically retrieved from the OKB. Once the selection is locked, the user needs to select or create annotation data. If an existing OWL individual is available from the OKB, the user can select it and then accept it as it is or modify its contents. Otherwise, if there are no OWL individuals available, the user can enable the second dialog to create an OWL individual if an OWL object property is selected, and the state returns to where annotation type selection is made. In the latter case, if a data annotation is about to be made, then the data value (and the corresponding unit where appropriate) can be entered in another pop-up dialog, in the state of "fill data annotation". The states in bold border including the initial state indicate where dialogs are launched.

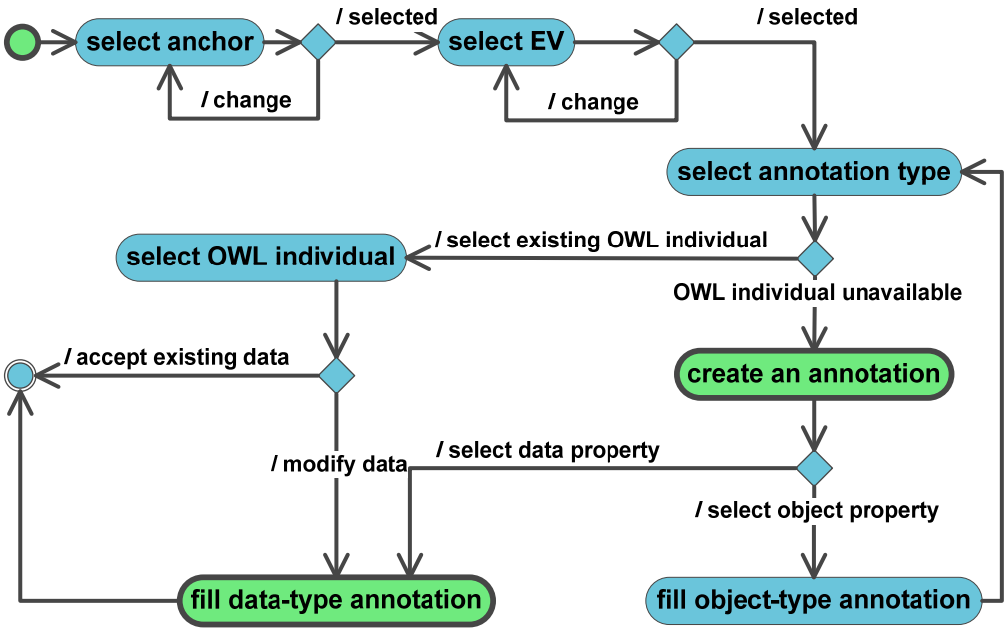


Figure 64 UML Flowchart for CAD Utility – Add Annotation

WBS 2.3.2 Ontology Utilities

As implied in Figure 61 and Figure 64, access to ontology and manipulation are needed from time to time, and many of the activities are repeated. Therefore, the subtask WBS 2.3.2 aimed to build a set of ontology utilities by using the OWL API, thus easing the programming work in other components. As illustrated in Table 39, this set of utilities

includes open and closed ontologies, class and data queries in various situations, reasoning, processing AW and so on. The logic for most of these is further described as pseudo codes from Figure 65, to Figure 74. Some utilities have similar algorithms, therefore are illustrated with common pseudo codes, such as Figure 65 and Figure 71.

Table 39 Key Methods in the JAVA Package – Ontology Utilities

Ontology Utilities	Description	Logic
createDataPropertyTriple() createObjectPropertyTriple()	Create an OWL axiom that represents subject-predicate-object relation, in which the inputs are OWL class name, data/object property name and a data value/an individual respectively.	Figure 65
createIndividual()	Create an OWL individual as given name and assigns it as a given type of OWL class.	Figure 66
fillDataForAnnotationChain()	Construct a data annotation chain with given data value filled in.	Figure 67
getAssertedIndividualAll()	Walk through ontology and return a list of all individual names under a given class name.	Figure 68
getData()	Given an anchor and a query parameter (i.e. an OWL class type), return the data value of any instantiated individual of this type in regard to this particular anchor.	Figure 69
getRefinedEVList()	Return an inferred list of fillers and the corresponding object properties, which have association with the concerned EV based on reasoning process. It has been helpful to filter out irrelevant options for user to annotate when a certain EV context is selected.	Figure 70
getTabbedAssertedClassList()	Return a list of hierarchical asserted subclass names of a given class. This is mainly used by the dialogs when a hierarchical names of subclasses need to be displayed.	Figure 71
getTabbedInferredClassList()	Return a list of hierarchical inferred subclass names of a given class, based on reasoning process.	Figure 71
getAWIndividuals()	Print out and return a list of name pairs - AW name and satisfied individual name. Note: an AW is a class with conditions (a set of axioms) as engineering rules.	Figure 72
openOntology() closeOntology()	These methods load or unload the ontology.	
processAnnotationChains() processAnnotationChainNodes()	The method processAnnotationChains () traverses all object properties of a given individual. It processes if any object property filler is linked to further annotations by calling the method processAnnotationChainNodes ().	Figure 73
processDataProperties()	Return a set of all data properties in all super-classes of a given individual.	
processIndividualDirectProperties()	Return all data and object properties associated with a given individual.	
processQuery()	Return a list of data values according to the required parameter, target entity and the inferred ontology. The parameter is the query, e.g. "PRODUCT DESCRIPTION - Process", the target is the individual name being queried, e.g. a CAD geometric element “BODY_1”.	Figure 74
populateMass() populateVolume() populateArea() populatePerimeter()	Create annotation chains for CAD model based data (i.e. mass, volume, area, perimeter), and save into the OKB as instances associated with bodies or faces accordingly.	

```
Open ontology;
Create OWL individual for the named subject;
Create OWL data property for the named predicate;
Create OWL literal for the appropriate type of data value, e.g. double, integer or string.
Create OWL axiom using these three entities;
Apply all changes and close ontology.
```

Figure 65 Pseudo Codes for the OWL Data Property Assertion Axiom

```
Open ontology;

Get the OWL class C as the input name;
Create an OWL individual I as the input name;

Create an axiom to associate I as a type of C;
Apply all changes and close ontology.
```

Figure 66 Pseudo Codes for Creating an OWL Individual

```
Open ontology;

For each node in the given annotation chain until the last node
{
    Automatically generate an individual name based on the class name;
    Create this axiom, e.g. body_1 hasWeight weight_body_1;
}
If it is the last node in the chain
{
    Create a data type axiom with the given data value;
}
Apply all changes and close ontology.
```

Figure 67 Pseudo Codes for Building a Data Annotation Chain with a Given Data Value

```
Open ontology;
Get a list of all subclasses under a given class;
For each class in this list
{
     Get and save all names of its individuals in a list L;
}
Close ontology;
Return the name list L.
```

Figure 68 Pseudo Codes for Getting OWL Individual Names of a Given Class

```
Collect all object property axioms OPAs and data property axioms DPAs of a given class (query parameter associated with this anchor)

For each OPA
{
    Get and return the data value if this OPA has data property axioms;
}

Otherwise if the DPA is not empty
{
    Get the data value and return;
}
```

Figure 69 Pseudo Codes for Getting Data Value for a Query Associated with an Anchor

```
Open ontology;

Get target individual;
Get target EV;
Enable reasoner;
Get a list of inferred subclasses of this EV – target-EV-class-list;

Get a set of axioms – S for the class of the given target individual;
For each axiom in S
{
    Visit and collect all concerned properties in the current axiom;
    For each object properties in the axiom
    {
        If the target-EV-class-list contains the filler's class in the current property
        {
            Record the filler's class and their corresponding object property in a result list;
        }
    }
}
Close ontology;
Return the result list.
```

Figure 70 Pseudo Codes for Getting Refined Class List for an OWL Individual According to an EV Context

```
Open ontology;
(Enable the reasoner and perform reasoning action in the case of getting inferred class names ;)

Get a set S of all subclasses of a given class;
For each class of this set S
{
    Get all subclasses' names and save in a list L;
    Get current indents;
    Save the names with a prefix of current indents into a result list – RL;
    For each class in L
    {
        If it has further subclasses
        {
            Increase current indent;
            Call this method itself;
            Append the returned name list into RL;
        }
    }
}
Close ontology;
Return the name list RL.
```

Figure 71 Pseudo Codes for Getting Asserted or Inferred OWL Subclass Names of a Given Class in Tabbed Style Hierarchy

```
Open ontology;
Enable the reasoner and perform reasoning action;

Get all AWs and their names;
For each AW
{
    If it has any satisfied OWL individuals
    {
        Record the individual name and corresponding AW name as a pair in a result list RL;
    }
}
Close ontology;
Return the result list RL.
```

Figure 72 Pseudo Codes for Getting Names for All Satisfied AWs

```
Open ontology;
Get all classes of the given individual;
For each class
{
    Get all object property axioms and data property axioms in this class definition;
    For each object property axiom
    {
        Recursively process further until an end node is reached;
        Update and save the list for object type annotation chains;
    }

    For each data property axiom
    {
        Update and save the list for data type annotation chains;
    }
}
Close ontology;
```

Figure 73 Pseudo Codes for Processing Annotation Chains

```
Open ontology;
Get all object properties and their corresponding fillers Fs from the individual (Target);

Enable the reasoner and perform reasoning action;
Get all individuals (Candidates) of the being queried parameter including all instances from all
equivalent classes based on reasoning;

For each member of Candidates
{
    If Fs contains this member (implies this target has the information associated with the query)
    {
        Get all data values from annotation chains ACs under this member (see Figure 73);
        Save values in a result list RL;
    }
}
Close ontology;
Return the result list RL.
```

Figure 74 Pseudo Codes for Retrieving Data Values for a Specific Query

WBS 2.3.3 AW

As noted in Section 5.5.1, an AW is a named class with predefined conditions that represent engineering constraints. One application of an AW is to define a condition to monitor whether a dataset can be satisfied. Based on the aspect of individual membership in factual reasoning, any individual satisfying this AW will be identified by the reasoner. Therefore, all data associated with this individual in regard to the dataset can be retrieved and exported.

In this prototype OntoCAD system, a spreadsheet is chosen as a universal interface for

data exchange with downstream applications, as many tools take data from files in order to operate. The spreadsheet contains a list of parameters that need to be queried. In order to achieve automation of queries, an application – Transformation Agent (TA) was developed. In conjunction with the OGUI, it can read the spreadsheet, execute all queries automatically in turn, and populated with values back to the spreadsheet.

The AW in the experimental work was mainly focused on the case of cost estimation. The cost tool SEER-MFG supports a server mode (Galorath Incorporated 2005). In such mode, SEER-MFG can export a command spreadsheet according to a specific project, which contains all the parameters need to be filled in order to compute the costs, and then take this spreadsheet back in as input file to drive the computation. To save experimental effort, it is not necessary to fully implement the server model, but the spreadsheet is passively processed by TA. The TA can be invoked by clicking on the AW dialog by a user if an AW is satisfied. In the SEER-MFG case, a command spreadsheet (e.g. Table 40) will be fed into the OMA, and handled by the TA. Every parameter will be queried to the OKB through the OMA, and the concerned data will be retrieved and filled in the spreadsheet in appropriate cells.

Table 40 Example of a SEER-MFG Command Spreadsheet

Parameters	Value (Least)	Value (Likely)	Value (Most)
PRODUCT DESCRIPTION - Material			
PRODUCT DESCRIPTION - Raw Material Cost Per Kg.			
PRODUCT DESCRIPTION - Process			
PRODUCT DESCRIPTION - Finished Weight (kg)			

The process in the TA is illustrated in Figure 75. The state “open command file” in bold border indicates where a dialog pops up for users to select the command file. When iteratively making queries to the OKB, the method “processQuery ()” in the ontology utilities is called as described in Table 39 and Figure 74.

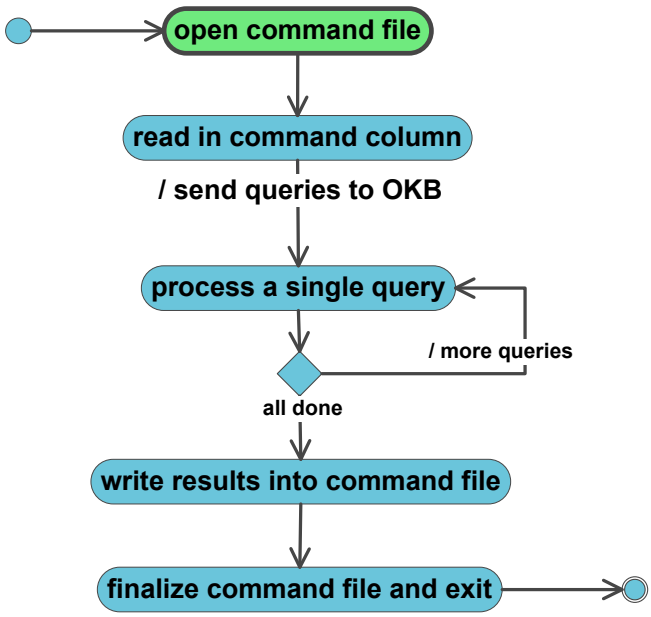


Figure 75 Process for the Transformation Agent to Retrieve Required Parameters

6.3.4 Implementation of Final Integration

After the OKB, OGUI and OMA were developed separately, these three components were integrated as a total system. As noted before, the OMA functions are invoked by user interaction with the OGUI, and performed by manipulating knowledge in the OKB, as the cooperation illustrated in Figure 54. This final integration was achieved by three steps.

The first one was to put the required callback functions in an appropriate place in the OGUI codes, for example, the Java object “OntoCADAutoLabel” needs to be called in the callback function “OntoCADAutoLabelCB”, which needs to be registered with the an OGUI dialog. Furthermore, since the object “OntoCADAutoLabel” is initialized, it is able to issue method calls to access the OKB.

The second step was to place all these files in correct directories in order to setup NX6. The NX6 add-on application directory has three sub-directories: application, startup and UDO. The dialog files, JAVA packages and pop-up Menuscript files are placed in the directory – application. The Menuscript files for the main menu bar go into the startup directory so that the top-level menu can be loaded as NX6 starts. The user defined objects (UDOs) allow users to add custom objects inside of NX6, but were not needed in this experimental work.

The third step was to debug the system as a whole and tested according to the FAT specification (see Appendix 2). The evaluation will be described in the next chapter. Some refinements were carried out from time to time. And finally the software implementation was documented.

6.4 Concluding Remarks

Based on the previous study and the development of an intended support, an actual support to improve the knowledge management in mechanical engineering has been described in this chapter. In consideration of the reality, including the research resource constraints, the actual support tends to evaluate the intended support in a reduced scale, simplified situation, and focuses on the evaluation of the core contribution.

The development actually went through an iterative software development process, but was described as a more conventional process using the waterfall approach, which has four phases: requirement specification, design, implementation, evaluation and documentation.

In requirement specification, the boundary of the actual support and the differences between intended support and the actually support were clarified. The differences mainly lie in the usability of the GUI, the completeness of the knowledge base and standards-compliance, and the complexity of rules for the MEV agent. And then a prototype software application as an add-on application to NX6 called the OntoCAD system was designed as three main modules the same as the intended support: OntoCAD Knowledge Base, OntoCAD Graphical User Interface, and OntoCAD MEV Agent.

The implementation was divided into a work breakdown structure with three main tasks and some further sub-tasks according to the three main modules. The application was programmed in JAVA. The OntoCAD Knowledge Base adopted the OWL API to realize ontology access. The OntoCAD Graphical User Interface was mainly built with the help of the NX6 OPEN API, Menuscript for creating menus, and NX OPEN User Interface Styler for creating custom dialogs. The OntoCAD MEV Agent collaborated with both the GUI and the knowledge base. Finally, the three modules were integrated as a total system, and installed for debugging, testing and evaluation.

The entire software application development was described in diagrams, tables of JAVA programming methods, and pseudo codes for the logic. For further details, please refer to the digital copy of the program source codes. This prototype at this stage was only verified (debugged) along the development, and has not yet been validated against the design requirements. Testing was not described here as the evaluation including both verification (testing) and validation will be described in the forthcoming chapter.

Chapter 7 Evaluation – Case Studies

This chapter describes the fourth stage of the DRM: the Descriptive Study II (DS-II), where evaluation is carried out and documented. Evaluation is a good practice and essential in a research or development process, as the effect of a design support is only an assumption until an evaluation (Blessing and Chakrabarti 2009). As a matter of fact, the development of a support is an innovation based on assumptions within the current situation, which is then changed to a new situation based on the introduction of the support. Therefore, the effect including either 'desired' or 'undesired' must be evaluated.

The proposed support is evaluated in three aspects: support evaluation, application evaluation and success evaluation. The support evaluation (verification) was continually and iteratively carried out during the development of the support in the DRM stage – PS. It is the process of testing and debugging to verify whether the actual support (i.e. the prototype OntoCAD system) is programmed as designed. The support evaluation will be very briefly implied without highlighting as it is more an implementation issue than a research contribution. For example, if a dialog is correctly displayed, it implies the prototype is correctly programmed.

The application evaluation (validation) is to validate the applicability and usability of the support against the desired values of the key factors. This evaluation answers the questions including whether the proposed solution is applicable, whether it address the key factors and as expected. The application evaluation is a prerequisite to success evaluation, which has been carried out in the literature research and preliminary experimental work as a pilot study described in Chapter 1 to Chapter 4. It demonstrated the potential of the reviewed approaches and technologies in supporting engineering design.

The success evaluation is to validate the usefulness of the support. This tries to answer the questions including how successful the support is in satisfying measurable success criteria and whether all measurable success criteria are covered. It is the actual demonstration through the prototype system which projecting the intended support. Once measurable success criteria are validated against, the applicability and usability can be once again be demonstrated, thus supporting the success evaluation in this DRM stage.

In this chapter, an evaluation plan will be defined based on previously raised research questions and hypotheses. The two case studies will be described with emphasis on some key evaluation points. Apart from using the knowledge base, the knowledge

modelling methodologies will be also evaluated. It should be noted that OntoCAD in this chapter particularly refers to the OntoCAD prototype system for conciseness unless explicitly stated elsewhere. Finally, this chapter will be concluded with key findings from this evaluation process.

7.1 Evaluation Plan

To produce an evaluation plan, it is necessary to revisit the research questions, hypotheses and evaluation criteria raised in Chapter 4. In Table 41, the hypotheses H1 to H3 were made to address research questions Q1 to Q3, while further questions were also made in order to support the hypotheses. With regard to these questions and hypotheses, the evaluation criteria have been initially defined in Table 42. Based on these previous research outcomes and considering the intended support and the developed actual support, an evaluation outline can be produced, which can be expanded to derive an evaluation plan.

Table 41 Revisit of Research Questions and Hypotheses

Research Questions		
Q1: How can knowledge be captured and represented to aid CAD system? Q2: How can knowledge and information interoperate? Q3: How can engineering services/tools be integrated with CAD system?		
Hypotheses	**Further Questions**	
H1: Annotation can be used as a mechanism to capture knowledge and as a media to represent knowledge while maintaining associations among information entities.	H1-Q1	How can annotation be used to capture knowledge?
	H1-Q2	How can annotation be used to represent knowledge?
	H1-Q3	How is the association maintained?
	H1-Q4	How annotation technologies have been used in engineering field and what are the weaknesses of current applications?
H2: Ontology can be used as an approach to construct, control, manage and process semantics so that to aid engineering design by incorporating heterogeneous engineering expertises.	H2-Q1	How can ontologies be used to define semantics?
	H2-Q2	How can ontologies possibly be used to manage semantics in respect of incorporating various expertises?
	H2-Q3	How can ontologies possibly be used to process the defined semantics?
	H2-Q4	How have ontological technologies been used in the engineering field and what are the weaknesses of current applications?
H3: The combination of annotation and ontology may support KIM by providing a platform for CAD systems to incorporate other engineering services/tools.	H3-Q1	How can annotation and ontology respectively aid CAD systems?
	H3-Q2	Whether annotation and ontology complement each other? If yes, how can annotation and ontology be combined; otherwise whether there is alternative?

Table 42 Success Criteria and Measurable Success Criteria

Success Criteria	Measurable Success Criteria
SC1 Knowledge can be acquired.	MSC 1. Knowledge can be captured through CAD system.
SC2 Knowledge can be stored.	MSC 2 Knowledge can be formally specified and saved in a knowledge base.
SC3 Knowledge can be represented.	MSC 3 Knowledge can be represented through CAD system.
SC4 Knowledge can be associated with a design model.	MSC 4 Knowledge can be associated with CAD model at different levels of granularity.
SC5 Knowledge can be shared.	MSC 5a Knowledge can be retrieved from knowledge base.
	MSC 5b Knowledge can be shared within the total system.
SC6 Data can interoperate.	MSC 6 Knowledge data can be exchanged with external systems.
SC7 System can be extended.	MSC 7 System interface is adaptive, and downstream tools can be integrated with few resources required.
SC8 Knowledge process can be automated.	MSC 8 Automation of knowledge processes can be achieved.

7.1.1 Evaluation Outline

The evaluation outline (EO) is defined in three major divisions in order to fully evaluate against the MSC (Table 42) based on the two case studies: cost estimation and FEA.

EO 1 With regard to MSC 1, OntoCAD needs to show the ability to capture user inputs or extract geometric information as annotations (data) and EO 2.

EO 2 With regard to MSC 2, OntoCAD needs to show the ability to store the captured annotation data as ontology entities (knowledge) into the OKB.

EO 3 With regard to MSC 3, OntoCAD needs to demonstrate the ability to represent annotations according to a specific request.

EO 4 With regard to MSC 4, OntoCAD needs to demonstrate the ability to annotate CAD models at least three levels of anchoring granularity: edge, face and body. OntoCAD also needs to demonstrate the association between CAD model and the OKB (conceptual reasoning), and also to demonstrate the persistence of this association.

EO 5 With regard to MSC 5:

a. OntoCAD needs to demonstrate semantic retrieval (conceptual and methodological reasoning), which means a query can be made and satisfied with explicit and correct information by traversing the knowledge base.

b. OntoCAD needs to demonstrate that knowledge of an EV can be shared by another EV.

EO 6 With regard to MSC 6, OntoCAD needs to demonstrate the knowledge retrieved as in EO 5 can be exported to external systems. Theoretically, the intended support can lend the knowledge in one system to another.

EO 7 With regard to MSC 7:

a. OntoCAD needs to demonstrate the system is configurable by modifying ontologies.

b. OntoCAD needs to demonstrate that few or even no programming changes are required to reflect knowledge base modifications.

c. OntoCAD is a general approach, which is extendable by modifying or incorporating more EVs or applications to adapt to environment changes.

EO 8 With regard to MSC 8, OntoCAD needs to demonstrate the automation process by automatic reasoning actions over some query results. This can be evaluation on AWs for a specific tool/service.

Using this EO as a set of evaluation objectives, two case studies have been carried out and evaluated according to a detailed evaluation process defined in the document – OntoCAD Functionality Acceptance Test Specification (Appendix 2). Based on the two cases, the knowledge modelling methodology was additionally evaluated. In the rest of this chapter, only key objectives closely related to the EO are described. For other operational details please refer to the FAT specification (Appendix 2), which is a guide for full evaluation process, also can be used as a user manual to understand how to operate this OntoCAD prototype system.

7.2 Case Study – Cost Estimation

As described in Section 6.2 and Section 6.3 for design and implementation of OntoCAD, the first experimental case – cost estimation, is to integrate the conventionally uncoupled costing tool SEER-MFG with a CAD system NX6. The SEER-MFG tool needs parameters

as inputs including dimensions/volume, materials, geometric features and so on to calculate cost results. Therefore ontologies for the cost EV and SEER-MFG tool need to be integrated into the OKB and then populated with data, which has been done and described in Chapter 6. There are two methods to feed in parameters: either manually import a command file that contains all required parameters or programmatically load the command file through command line instructions (i.e. server mode). The first interactive method was adopted in this evaluation process to simulate the automatic server mode.

The interactions among end users, OntoCAD and SEER-MFG are as shown in Figure 76 (identical to Figure 46 in Chapter 6). In this use case, when a cost analysis request is initiated by a user, SEER-MFG sends a request to the OntoCAD system, in which the OMA makes a judgement whether current data is sufficient for SEER-MFG to operate. If not, the agent asks the user to provide more information. Meanwhile, the agent assigns an AW to observe the status of OKB for when data becomes sufficient. Once sufficient data become available, the OntoCAD agent will export the data to SEER-MFG to compute and then return a real-time result to the user. The following sub-sections will describe how key evaluation procedures have been done to demonstrate against the EO.

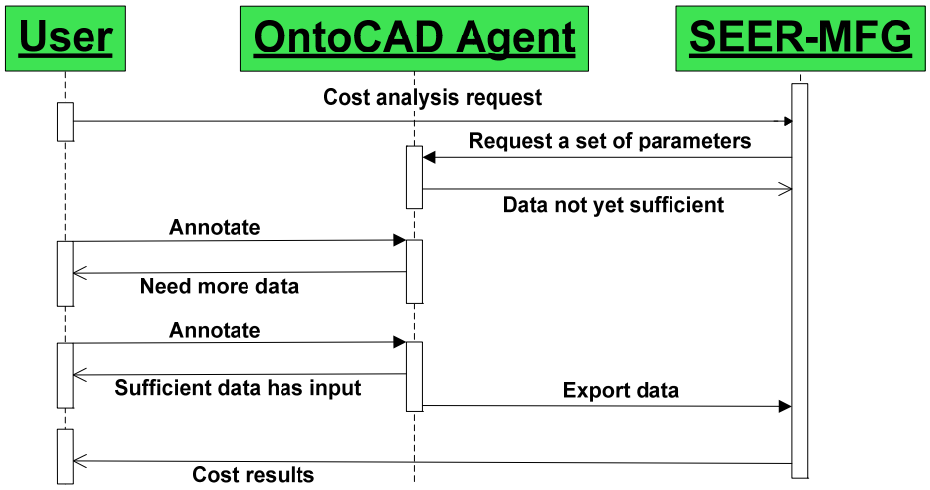


Figure 76 Sequence Diagram for Integration of NX6 and SEER-MFG (repeat of Figure 46)

7.2.1 Evaluation on EO 1 and EO 2

This section is to evaluate whether OntoCAD can manually capture user inputs and automatically extract geometric information from CAD model as annotations (EO 1). This task also closely associates with evaluating whether annotation data can be stored (EO 2). The operations are as following:

1) Run NX6 to load the CAD model – towbar and enable OntoCAD[12]. Run "Auto Label" function (Figure 77) and check the following:

   a) Check if all edges, faces and the body in the towbar part are labelled.

   b) Check with Protégé if all corresponding individuals are created in the OKB, including geometric entities and automatically populated individuals shown in Table 43.

2) Start "Add Annotation" function, and annotate the body with a manufacturing process sand casting.

   a) Check that BODY_1 is shown in the dialog "Add OntoCAD annotation" when selected (Figure 78).

   b) Check that when the EV concept 'cost driver' is selected (Figure 79), only corresponding cost related factors are available in the dialog (Figure 80).

   c) Check that an individual of class "SandCasting" can be created (Figure 81 and Figure 82), filled with a text string "Sand Casting" (Figure 83).

   d) In Protégé editor, check that BODY_1 of geometric item type manifold_solid_brep is associated with object property assertion "hasManufacturingProcess SandCasting_1". And check that SandCasting_1 has chained indirect annotations with a string value "Sand Casting" in the last node as shown in Figure 84.

Table 43 Examples of Automatically Populated Instances from CAD Model

Geometric Entities	Populated Instances
Body	Area, mass, volume, weight
Face	Area, perimeter
Edge	Length

[12] To enable OntoCAD prototype system, click on "Start" button on the NX6 toolbar, and then "All Applications", and then click on "OntoCAD" at the bottom of the drop-down menu.

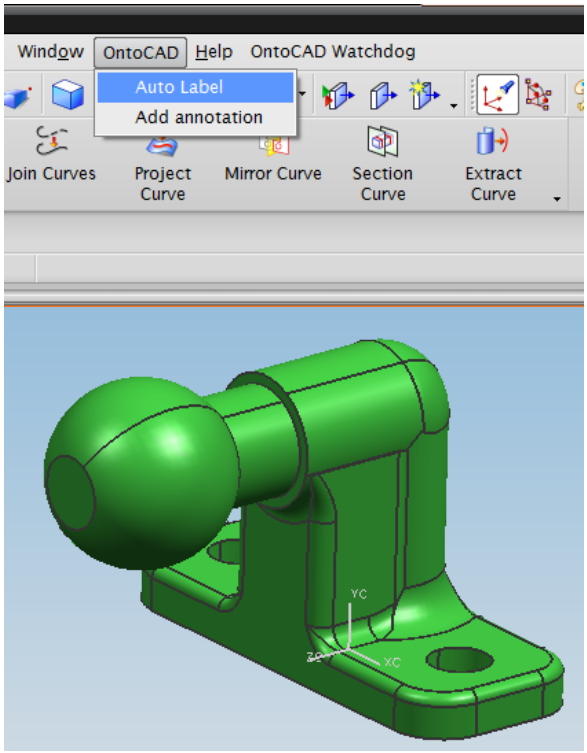


Figure 77 Automatic Generation of Annotation Anchors

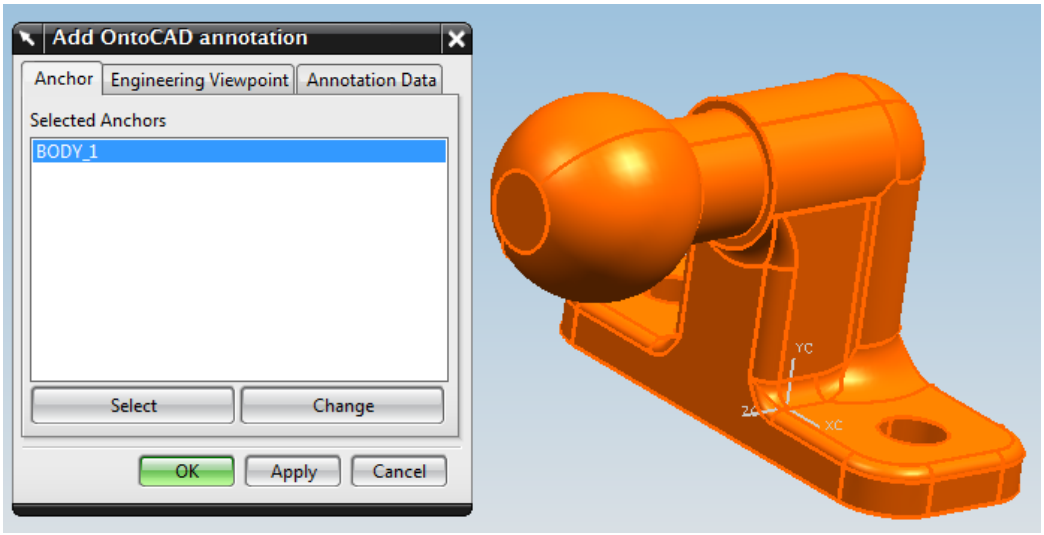


Figure 78 A Selected Anchor Shown in Dialog

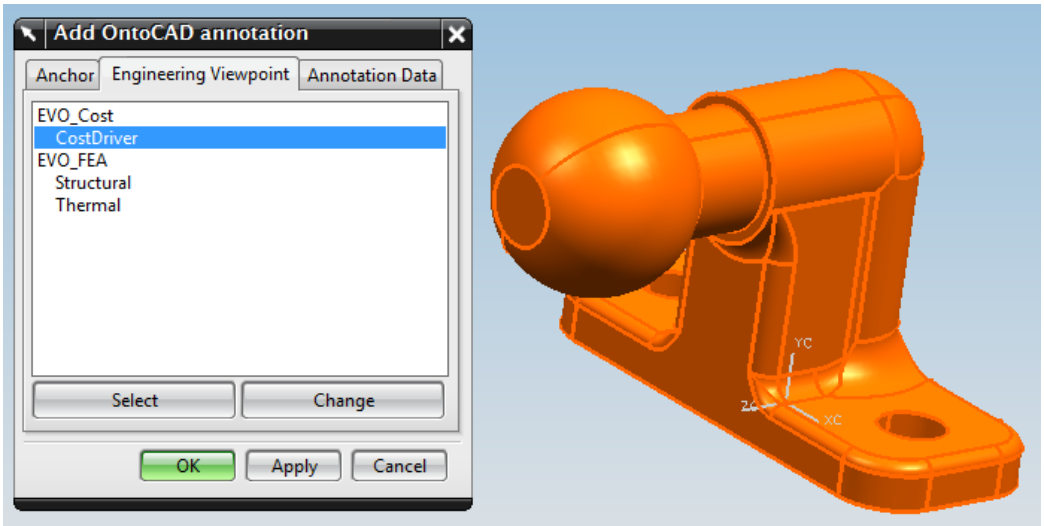


Figure 79 Selecting an Engineering Viewpoint

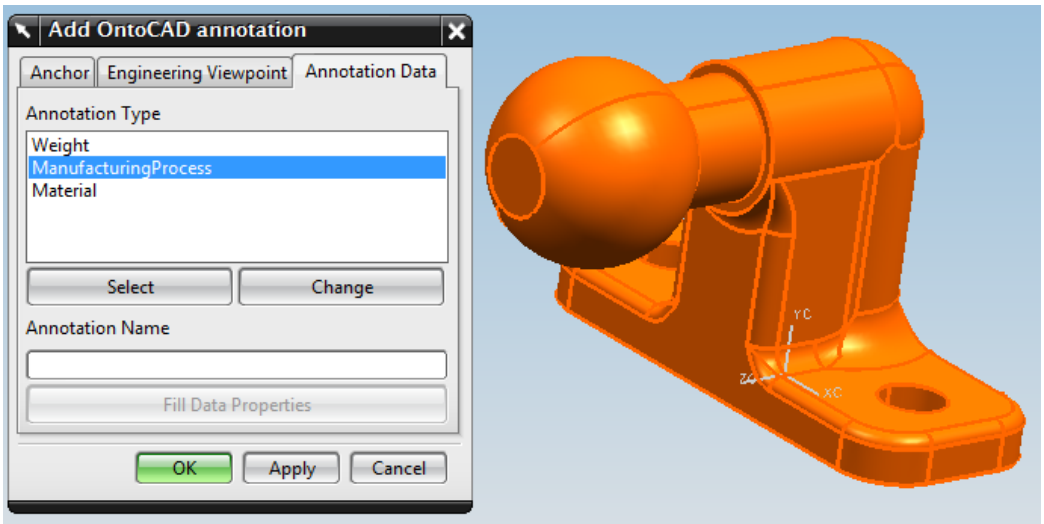


Figure 80 Available Annotation Types With Regard To a Specific Engineering Viewpoint Selection

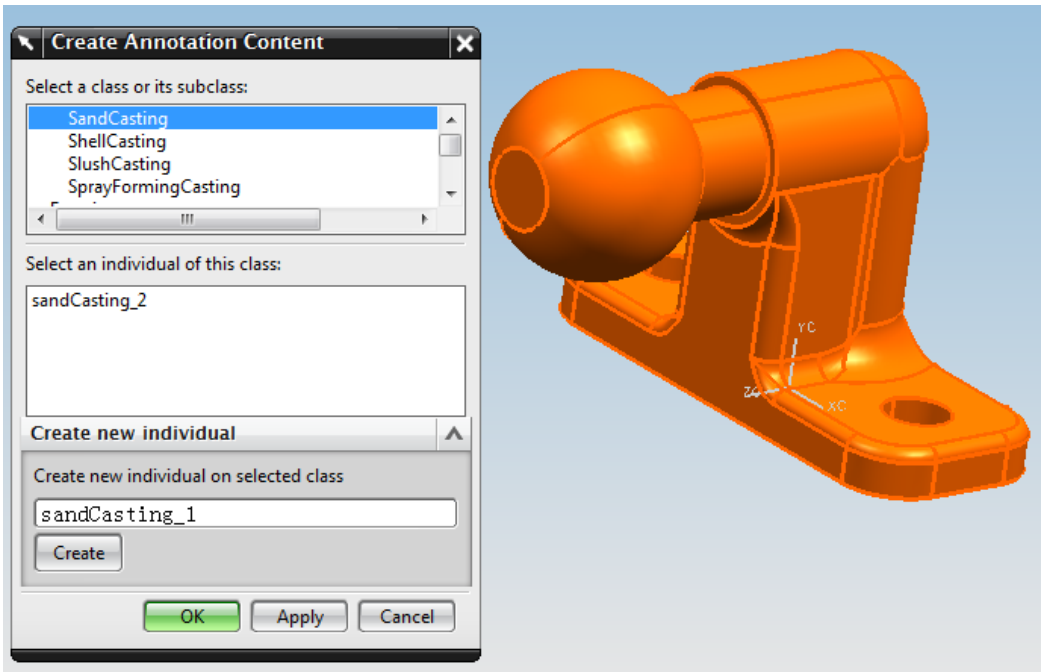


Figure 81 Creating a New Individual as Annotation Content

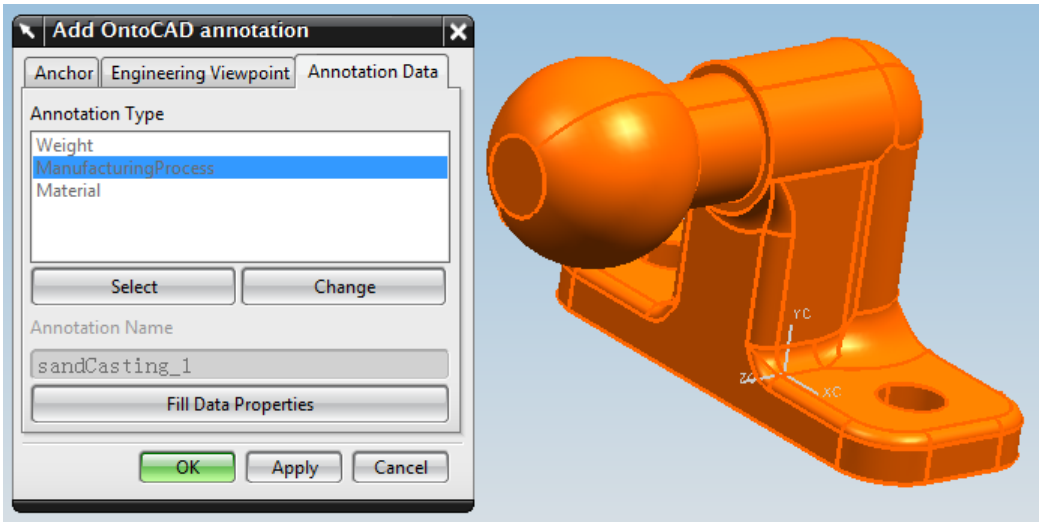
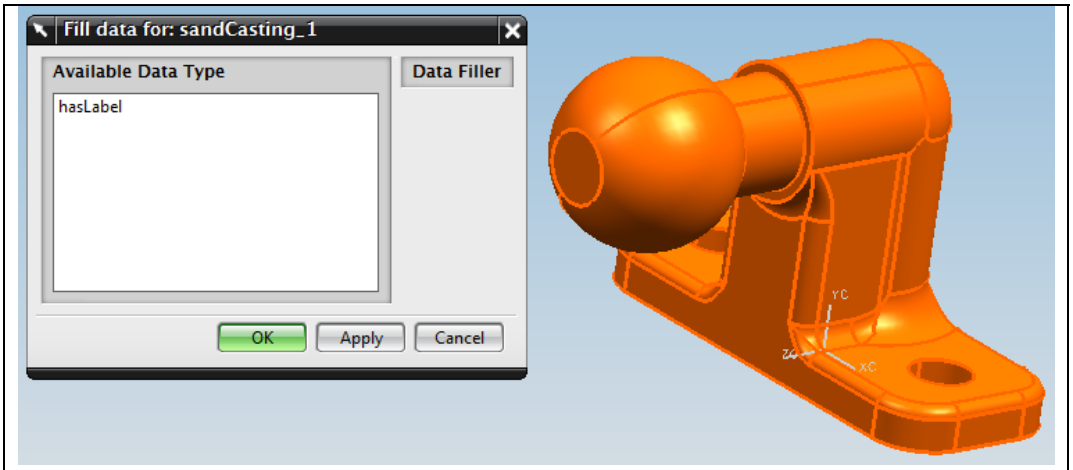
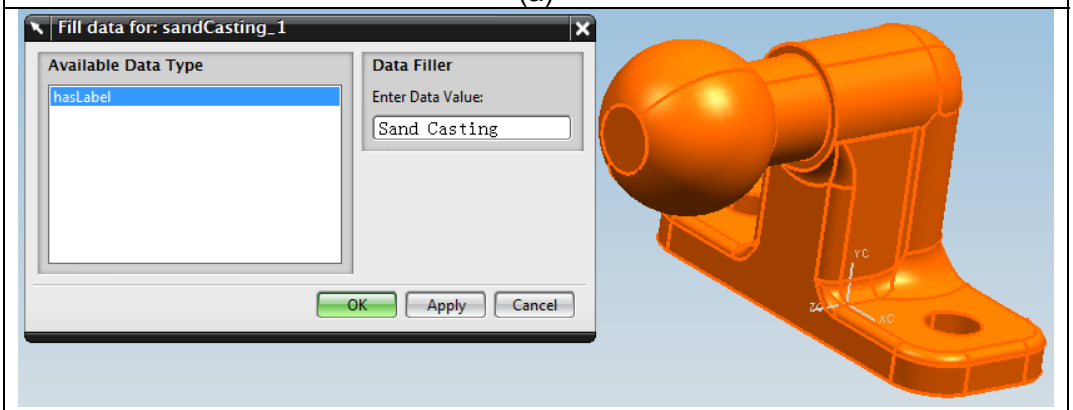


Figure 82 An Annotation Entity Is Locked After a Selection or Creation



(a)



(b)

Figure 83 Fill Data Value with Appropriate Unit

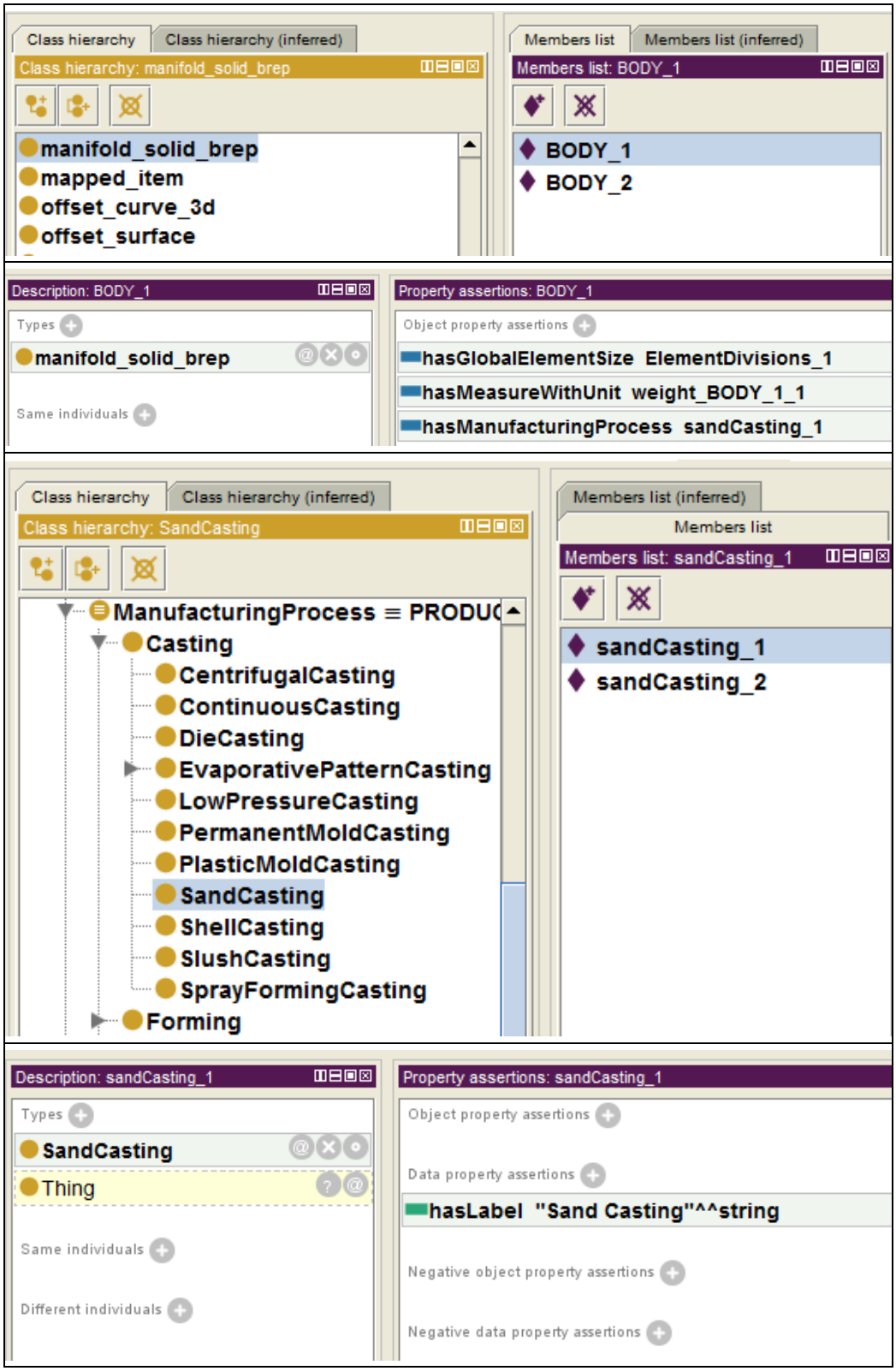


Figure 84 Example of Created Annotation Data in the View of Protégé

7.2.2 Evaluation on EO 3

To demonstrate representation of annotations previously captured, it can be implied by the step 2) b) in Section 7.2.1 (annotation data selection). The already created weight

entity weight_BODY_1_1[13] can be shown and selected, which implies annotation data can be dynamically retrieved as specific request according to the context.

EO 3 can also be demonstrated by displaying content of a specific annotation. The operations are as following:

1) Repeat the procedures until step 2) c) described in Section 7.2.1.

2) Click on the button "Display" as shown in Figure 82. Check that the annotation content of weight_BODY_1_1 including data value and the associated unit is displayed in a dialog (Figure 85).

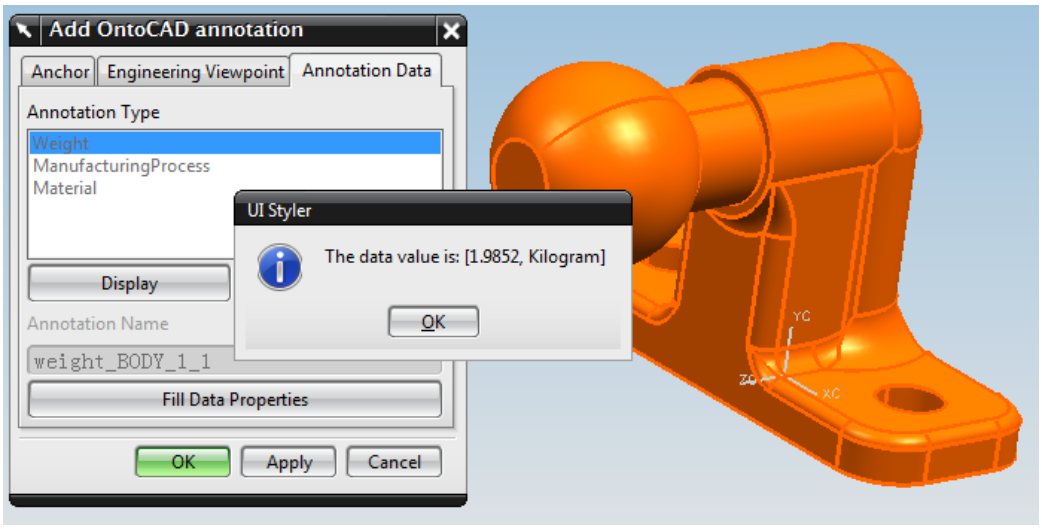


Figure 85 Example of Displaying Annotation Content

7.2.3 Evaluation on EO 4

Firstly, to demonstrate three levels of anchoring granularity G1 (body), G2 (face or faces) and G3 (edge or edges), there are three set of procedures:

1) Repeat the procedures in Section 7.2.1 for G1.

2) Repeat the procedures in Section 7.2.1 for G2 instead. The difference is adding a manufacturing process – grinding to the tow ball surface, and check that corresponding OWL entities are created.

3) Repeat the procedures in Section 7.2.1 for G3 instead. The difference is adding a manufacturing process – drilling to the edge of one hole, and check that

[13] This is an instance of weight associated with a geometric entity 'Body_1', which is automatically extracted and populated from the CAD model into the OKB at background.

corresponding OWL entities are created.

Apart from demonstrating the association between annotation data and geometric entities, a demonstration for persistent anchoring has also been attempted in the case of static CAD model. Persistent anchoring of dynamic models has not been considered. This needs the following procedure:

1) Save the towbar part with NX after automatic anchor generation has been executed. Export this part as a STEP-203 file. Open the STEP file with NX on another computer and check that all labels are shown.

2) Run application CATIA and Autodesk Inventor in turn to import this STEP file. Check that all concerned geometric items are correspondingly labelled.

This is to verify the assumption that the anchoring mechanism is robust if an NX6 part with labels can be exported into STEP file and can be imported and recognized by other CAD systems, including NX, CATIA® and Autodesk Inventor. Unfortunately, only NX passed the test. The other two CAD systems recognized the labels for bodies, but ignored other levels. This result implies that this STEP-compliant anchoring mechanism has the potential of persistent anchoring, but has limited control over the implementation of CAD application vendors.

7.2.4 Evaluation on EO 5

The evaluation EO 5a demonstrates semantic retrieval in two aspects: conceptual reasoning and methodological reasoning. Semantic retrieval based on conceptual reasoning has been demonstrated by the procedures in Section 7.2.1, in which a dynamic list of cost drivers (classes) is retrieved from the OKB after an EV is appointed.

In the second aspect, methodological reasoning can be demonstrated with an AW evaluation. Data can be retrieved not only based on the asserted classes, but also based on inferred classes (i.e. class subsumption). This is the background process for knowledge mapping between equivalent classes (equivalency checking). For example, to traverse the OKB to find any equivalent classes in relation to the parameters listed in Table 44, which the procedures described in Section 7.2.5.

Through this procedure, knowledge sharing between EVs (EO 5b) can be demonstrated since the knowledge of manufacturing processes has been used in the cost analysis. This is based on an assumption that the manufacturing process information can be automatically retrieved from a manufacturing process planning tool or manually entered

by users as described in Step 2) described in Section 7.2.1, such as attaching manufacturing process information.

Other than the EV manufacturing process, material and its properties are also EVs, in which knowledge needs to be shared by cost estimation. Related to manufacturing process, geometric information can also be annotated and used by cost estimation, for example the number of holes, or weight of a design part.

7.2.5 Evaluation on EO 6 and EO 8

This is to evaluate that knowledge can be retrieved and exported to external systems, in which way the ability of tool integration can be illustrated. This evaluation is demonstrated through three steps in the use case of SEER-MFG. The first step is to detect state triggering by an AW for SEER-MFG. The judgement is made by the OMA reasoner based on the AW rule. The second step is to feed in a prepared command file by a user. A partial file (Table 44, identical to Table 40) containing four cost drivers is used in this evaluation. The last step is to recursively retrieve all required parameters and populate into the given command file. The procedures are as following:

1) To define an AW that monitors the status of a set of parameters (Table 44) as illustrated in Figure 86.

2) To repeat the procedures in Section 7.2.1 to annotate BODY_1 about material, material property, manufacturing process. Note: there are also data which have been automatically populated from CAD model into the OKB, such as weight. The background process is also a demonstration of knowledge sharing (Section 7.2.4).

3) Check that if SEER-MFG AW for BODY_1 is available in the dialog of application watchdogs (Figure 87 a) during annotating. Note: this is an implication of process automation.

4) If AW for BODY_1 is available, select the command file (Figure 87 b) and export this dataset (Figure 87 c). The background actions include continually making four queries for the listed parameters and retrieving the correct data and fill in the appointed command file as described in WBS 2.3.3 in Chapter 6. Open the command file and check that correct data have been entered.

This evaluation implies that further automation can be achieved so that downstream tools can be more seamlessly integrated into OntoCAD as a total system. In this particular case, it is to automate the procedure in Step 4) by using the server working mode of SEER-MFG,

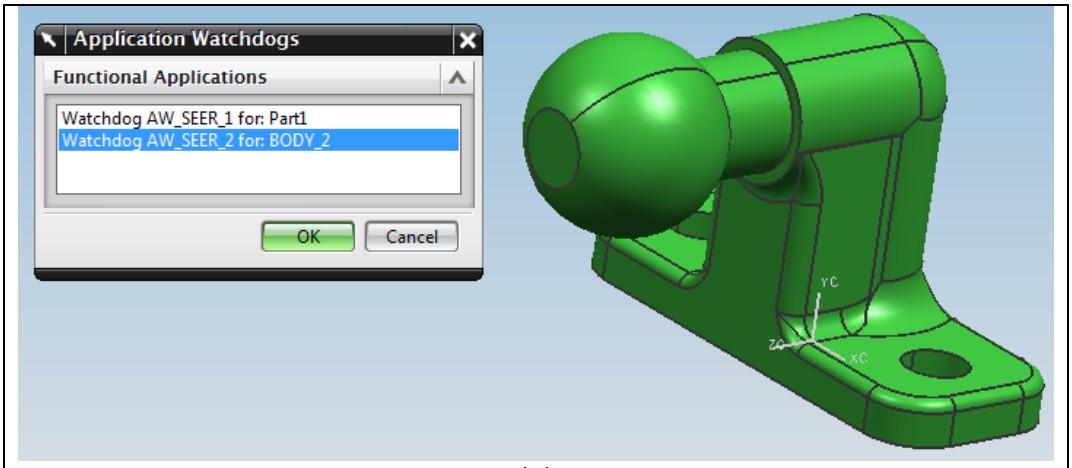
in which a command file can be automatic loaded by calling command line instruction function, and the cost results can be automatically extracted and fed back to OntoCAD users in a real-time way. In this way, SEER-MFG can provide a cost analysis service running in the background, thus enables more accurate cost estimating concurrently as a design object is developed. This automation feature has been demonstrated as part of the work of Newnes et al. (2007). To avoid unnecessary repetition of the previous work, this potential fine-tune feature on a particular downstream tool is not implemented within this research

Table 44 Example of a SEER-MFG Command Spreadsheet

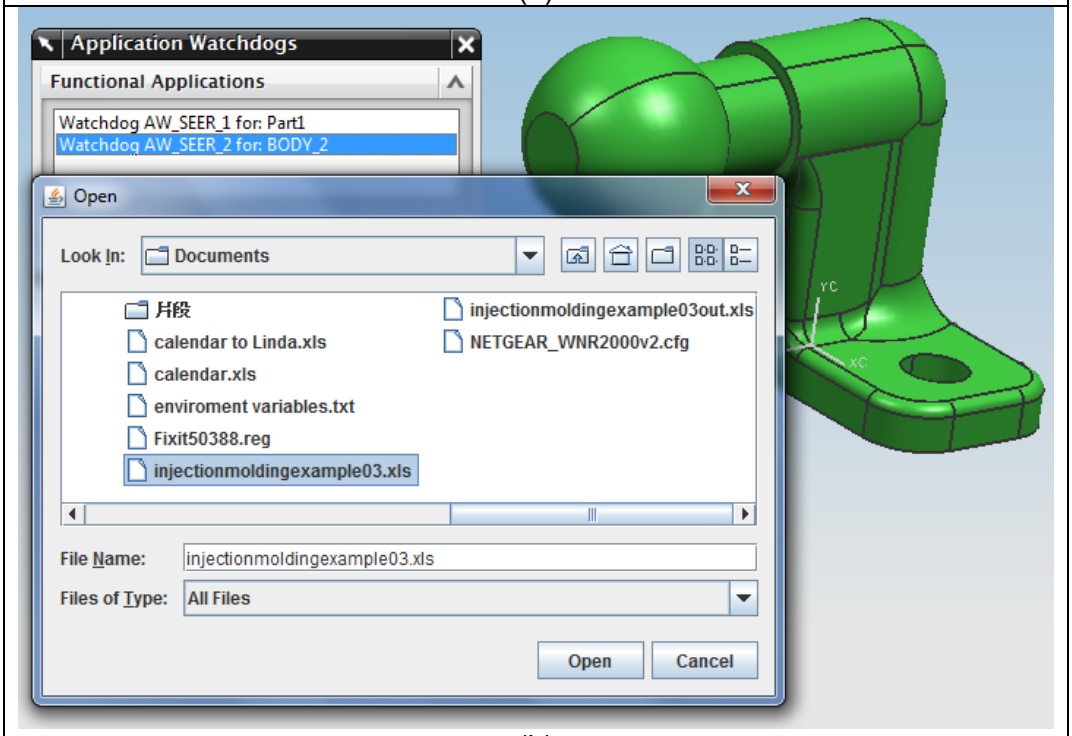
Parameters	**Value (least/likely/most)**		
PRODUCT DESCRIPTION - Material	Ductile cast irons		
PRODUCT DESCRIPTION - Raw Material Cost Per Kg.	0.6325		
PRODUCT DESCRIPTION - Process	Sand casting		
PRODUCT DESCRIPTION - Finished Weight (kg)	1.9852	1.9852	1.9852

(Part or geometric_representation_item)
and (hasManufacturingProcess some PRODUCT_DESCRIPTION_-_Process)
and (hasMaterial some PRODUCT_DESCRIPTION_-_Material)
and (hasMaterialProperty some
PRODUCT_DESCRIPTION_-_Raw_Material_Cost__Per_Kg.)
and (hasMeasureWithUnit some measure_with_unit)

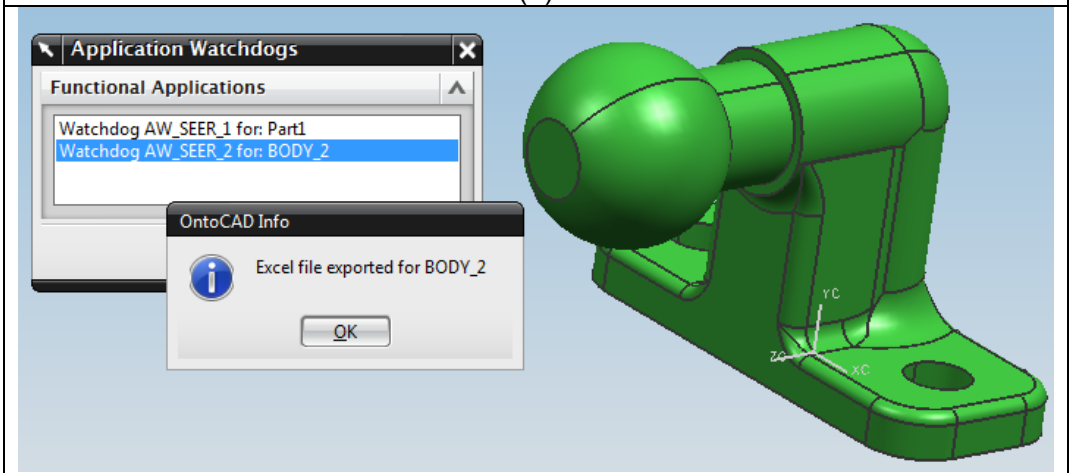
Figure 86 An Example Rule That Defines an AW for SEER-MFG



(a)



(b)



(c)

Figure 87 Application Watchdog Monitoring and Exporting Data to External Tools

7.2.6 Evaluation on EO 7

Two simple examples that demonstrate ontologies as configurations to drive OntoCAD (EO 7a and EO 7b) can be:

1) Dynamic GUI: repeat the procedures described in Section 7.2.1, but select various annotation data options, such as weight, material properties, manufacturing processes, and populate data. In the dialog of filling data (Figure 83), check that the GUI is adapted to user selections accordingly. For example, the user interface for unit selection options should be available for annotating weight (which has a numeric value), but not for manufacturing process (which is a string).

2) User context change:

   a) During the last procedure, check that the list in the tab “Annotation Data” is dynamically changed. This demonstrates the adaptive structure of annotation content based on the OKB.

   b) Modify the EVO class with Protégé by adding a new sub-class “EVO_Service”. Repeat the procedures to Step b) in Section 7.2.1. Check that a new viewpoint option appears in the list. Delete “EVO_Service” and repeat this procedure. Check that “EVO_Service” is not listed any more. This additionally demonstrates that no programming effort is needed as ontology is changed.

With regard to EO 7c, it is to demonstrate an overall effect by extending the system with a new EV. The second used case – FEA will be described in Section 7.3 for this purpose. In this case, only the OKB is updated with a new EVO. In contrast, programming modification for OntoCAD software application is not required.

7.2.7 Further Evaluation on EO 8

Demonstrations on some reasoning services have been described: conceptual reasoning in Section 7.2.3 and Section 7.2.6; conceptual and methodological reasoning in Section 7.2.4; factual reasoning for identifying instances for an AW, conceptual reasoning for equivalency checking and methodological reasoning for data retrieval in Section 7.2.5.

This section additionally evaluates methodological reasoning over the knowledge base, in which way the automated process on viewpoint-based engineering constraints can be illustrated. This evaluation is demonstrated through three steps in a general use case based on a manufacturing viewpoint. The first step is to define an AW representing a

constraint on sand casting and corresponding advice, as illustrated in Figure 88. This rule implies that it should be cautioned if any body entity has a manufacturing process as sand casting and also has a weight value either greater than 100 kilograms or less than 0.2 kilograms. The second step is to detect state triggering by the OMA reasoner according to this AW. And then the OMA gives the user advice on further action. The procedures are as follows:

1) To define an AW using Protégé that monitors the engineering constraint as shown in Figure 88.

2) To repeat the procedures in Section 7.2.1 to add annotations to BODY_1 (a G1 anchor) for manufacturing process – sand casting. The weight has been automatically populated from the CAD model (i.e. 1.9 kilograms). Check that if BODY_1 is not a member of this AW in the AW dialog. This is to evaluate the state change is not triggered as the weight of 1.9 kilograms is appropriate.

3) Load the ontologies with Protégé and manually change the weight value to 0.15 kilograms. Check that if BODY_1 becomes a member of this AW in the AW dialog. This is due to this new value is out of range for sand casting process.

4) Click on this available entry in the AW dialog. Check that if an advice message (according to Figure 88) is displayed in a pop-up window as illustrated in Figure 89.

This evaluation implies that further automation based on general use based on an EV, rather than a particular downstream tool. This feature opens the possibility for application developers to program further automatic operations.

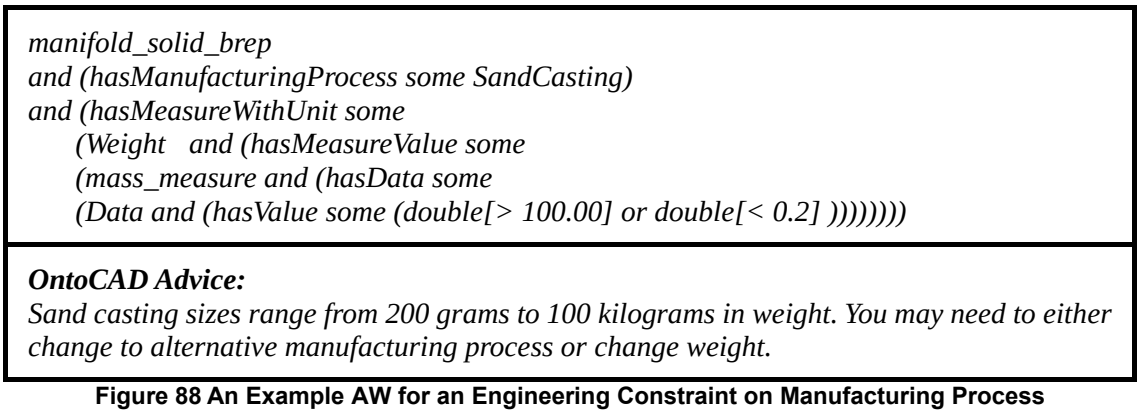


Figure 88 An Example AW for an Engineering Constraint on Manufacturing Process

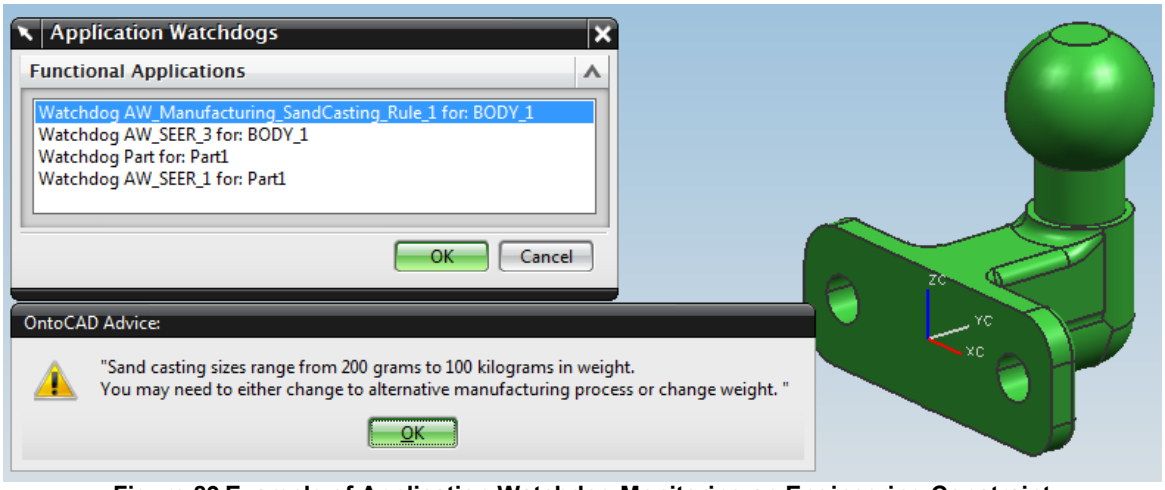


Figure 89 Example of Application Watchdog Monitoring an Engineering Constraint

7.3 Case Study – Finite Element Analysis

As noted before in Chapter 6, the use case of FEA is very similar to the cost estimation case, but mainly demonstrates generality and efficiency of the ontology modelling methodology again in the case of integrating new AO and EVO into existing ontologies. This case is more focused on incorporating knowledge of legacy tool rather than integrating a complex system into the OntoCAD paradigm.

Table 45 Exemplary Classes in Relation to FEA Viewpoint

FEA Related Class (# of Subclasses)	Exemplary Subclasses (# of Subclasses)
AO_ANSYS (6)	ConstantValue (0) DOF (12) ElementType (0) ElementTypeOption (3) Loads (5) Meshing (1)
EVO_FEA (2)	Structural (0) Thermal (0)
EVO_Design (1)	Note (0)

This FEA case simulates transferring a knowledge model from a tightly embedded CAD system or FEA system into an independent knowledge base, and integrating an EV into the OKB and allowing customized knowledge expansion, e.g. adding user interpretations. A generic FEA works through three stages: pre-processing, solution and post-processing stage. The decision and input data are needed from the analyst to prepare the finite element model in the pre-processing stage, such as meshing instructions, boundary conditions, initial conditions, and loading (Table 45). At the post-processing stage, interpreting of the analysis result may need to be recorded and referring to the CAD

model. In this way, the knowledge can be captured, stored, retrieved and reused by any system understand the OKB, rather than trying to understand every tool/service involved.

7.3.1 Evaluation on EO 7c

This FEA case study is mainly an application evaluation that complements the first case study to demonstrate the generality through applicability and feasibility. An AO for ANSYS® has been modelled and incorporated into OKB described in Chapter 6, in which terms (Table 45) used in ANSYS® are defined. The following procedures demonstrate that input data to the ANSYS® pre-processor can be captured and stored into the OKB as ontological annotations, where the towbar example is used again for stress analysis:

1) Run NX to load the CAD model – towbar and enable OntoCAD. Run "Auto Label" function if anchors have not been applied (see procedures in Section 7.2.1).

2) Start "Add Annotation" function, and annotate the body with a mesh size[14] of 12.

Other than preparing mesh, annotating loads and boundary conditions can also be demonstrated.

3) Start "Add Annotation" function, select the face of the towbar base, and annotate this face with a constraint of displacement[15] – fix the towbar base area.

This was successfully demonstrated that AO and EVO can be added to the knowledge base. To further demonstrate this point, terms in AO_ANSYS have been matched with the pre-processor of NX6 through visual inspection. One example inspection procedure can be:

4) Launch design simulation function in NX6, and select BODY_1 and 3D Tetrahedral Mesh. Check that there is equivalent configuration – Element Size as Step 2).

This demonstrates that at least part of FEA model with pre-processing definitions can be recorded in OKB, and theoretically this knowledge can be shared. In addition, NX6 supports importation and exportation with ANSYS® (Siemens PLM Software Inc 2008), which implies that a full FEA model can be potentially be exchanged through OKB between two systems.

---

[14] Operation hints: select BODY_1 -> select FEAViewpoint -> Structural -> Meshing > Size Controls -> Global Element Size -> Element Edge Length -> 12.

[15] Operation hints: select FACE_x -> select FEAViewpoint -> Structural -> Loads -> Displacement: hasDOF ALL_DOF; and hasData 0.

5) Furthermore, any legitimate anchors can be selected for users to add free-style comments[16]. This can be arbitrary text notes in natural language, or a reference (e.g. URI) to an external file in any format.

7.4 Knowledge Modelling Methodology

Due to the importance of knowledge modelling methodology and the hierarchical architecture of OKB, the OntoCAD knowledge modelling methodology is one of the core contributions that need to be demonstrated. This has been evaluated by comparing the modelling processes in the development of knowledge modules for simulating different scenarios (i.e. cost estimation and FEA).

As described in Section 6.3.1, the FO was modelled by following the middle-out route; EVO_Cost was modelled by following the middle-out route; other two EVOs for material and manufacturing were also developed as needed by EVO_Cost by following the middle-out route; the AO_SEER was modelled by following the bottom-up route; the EVO_FEA and AO_ANSYS were modelled by following the bottom-up route.

Time consumed by modelling corresponding ontologies is compared in Table 46. The FO has the longest development duration, due to the complexity of the STEP standards and the learning curve required. The EVO_Cost has the second longest duration for development and a long duration for data collection – Delphi method, in which the two-round questionnaires took 8 weeks, including questionnaire design, distribution and processing feedbacks. AO_SEER, AO_ANSYS and EVO_FEA have relatively shorter development time, for the reasons of gradually improved proficiency and relatively lower complexity than FO. According to the statistics in Table 46, the average efficiency of using different strategies for knowledge modelling can be calculated as Table 47. It is difficult to conclude which strategy is more efficient or advanced. It is more dependent on the situation of the specific case and the complexity of the ontology being developed.

[16] Operation hints: select BODY_1 -> select FEAViewpoint -> Structural -> Annotation Data -> Comment.

Table 46 Comparison of Time Consumption for Bottom-Up and Middle-Out Strategies of OntoCAD Knowledge Modelling Methodology

Ontology	Strategy	Time Consumption	Participants	Entities
FO	Middle-out	Development: 4 weeks Literature study: 8 weeks	Project group.	Class: 161 Property: 29 Individual: 79 Class axioms: 186 Total: 455
EVO_Cost	Middle-out	Development: 2 weeks Questionnaire: 8 weeks	Project group; A group of academic interviewees.	Class: 18 Property: 7 Individual: 2 Class axioms: 17 Total: 44
EVO_ Material	Middle-out	Development: 0.5 week Literature study: 1 week	Project group.	Class: 30 Property: 12 Individual: 5 Class axioms: 26 Total: 73
EVO_ Manufacturing	Middle-out	Development: 0.5 week Literature study: 1 week	Project group.	Class: 25 Property: 9 Individual: 0 Class axioms: 18 Total: 52
AO_SEER	Bottom-up	Development: 1.5 weeks Literature study: 1 week	Project group.	Class: 6 Property: 1 Individual: 0 Class axioms: 5 Total:12
EVO_FEA AO_ANSYS	Bottom-up	Development: 2 weeks Literature study: 2 week	Project group.	Class: 26 Property: 20 Individual: 5 Class axioms: 32 Total: 83

Table 47 Average Efficiency for Knowledge Modelling Methodology Strategies

Strategy	Efficiency (Total entities / Time Consumption)
Middle-out	624 / 25 = 24.96 entities/week
Bottom-up	95 / 6.5 = 14.62 entities/week

7.5 Concluding remarks

In this chapter, an evaluation outline was defined based on the developed actual support, which was derived from the intended support by focusing on the core contributions and compromising according to resource constraints for this research work.

This chapter then described some key evaluation procedures to demonstrate against the evaluation outline. This was divided into three partitions: cost estimation, finite element analysis, and OntoCAD knowledge modelling methodology.

All aspects listed in the evaluation outline were implemented in the case of cost estimation from EO 1 to EO 8. The evaluation procedures demonstrated the fundamental functions of the OntoCAD prototype system including the ability to capture, store and represent annotations, and to associate the annotation with three levels of granularity. It also demonstrated some advanced features including semantic knowledge retrieval, knowledge sharing among engineering expertise. Further more, data exchange with an external system, tool integration, knowledge processing automation, and knowledge base extension were also evaluated.

The second case finite element analysis was carried out as a complement to the first use case in order to demonstrate the generality of such an approach by showing applicability and feasibility on another use case. And this case study provided support on evaluating the OntoCAD knowledge modelling methodology.

Both the bottom-up and the middle-out strategies of the OntoCAD knowledge modelling methodology were evaluated. The foundation ontology and the engineering viewpoint ontology for cost estimation was modelled through a middle-out strategy for a scenario that no such ontology exists and tries to aid general purposes in the concerned knowledge domain. It took the longest duration due to the questionnaires for data collection. On the other hand, the engineering viewpoint ontology for finite element analysis, and application ontologies for SEER-MFG and ANSYS® were modelled by following the bottom-up route. These ontologies reflect a scenario that there are existing ontologies or a specific tool/service ready to be incorporated into OntoCAD as a total system. Development through this strategy had a relatively shorter duration due to lower complexity of the ontologies.

Based on the evaluation results and reflecting to the impact model of the proposed OntoCAD framework and its measurable successful criteria, the research questions and hypotheses are answered. Further discussion will be expanded in the forthcoming chapter, where potential direction of future work is also suggested.

Chapter 8 Discussion

In previous chapters, an intended support – the OntoCAD system for improving knowledge and information management in the engineering domain - has been designed and described based on a background research. The focus of the activity was aimed at the later design stages after CAD systems are employed. An actual support – the OntoCAD prototype system - has also been developed accordingly to evaluate the research contribution and ascertain whether the research aim and objectives were achieved. This chapter continues with discussions on what has been learned from the research - some achievements and limitations in the presented work, including types of target media, annotation anchors, annotation content, knowledge modularity, knowledge sharing and reuse, and knowledge modelling methodology, the generality of this framework.

8.1 Annotation Anchors

As the foremost component of annotation data, anchors play the role that maintains association between the object being referenced (target object) and the annotation content. Therefore, anchors need to be persistent, and have broad coverage for granularities and various target object types

8.1.1 Persistent Anchoring

The persistent anchoring has been an issue since the beginning of this research. This research work was originally inspired by Davies' PhD research work (2008), in which features within one CAD system are used as anchors, and confront a great challenge to use captured annotations outside this CAD system. This difficulty was caused by the persistent identification of geometric entities, which were used as the reference to the annotation, namely the anchor. According to Davies' suggestion and implication of other researchers' work (El-Mehalawi and Allen Miller 2003b; El-Mehalawi and Allen Miller 2003a), geometric entities can be potentially identified through the widely accepted industrial STEP format. This is to name geometric entities in a certain CAD system, such as Siemens' NX, and then export the CAD model together with these geometric labels in STEP format. This relies on the fact that the "name" attributes of geometric entities defined in STEP 203 have the exact required capability to carry labels. According to our experimental observation, NX parts can be labelled and exported into STEP format, in which the labels survive. In reverse, a STEP file with labels can be identified by NX. Therefore, the presented work in this thesis uses labels of geometric entities in a CAD

model and corresponding instances in the STEP-compliant knowledge base as persistent anchoring mechanism.

Although this mechanism theoretically overcomes the issue with the help of STEP, there are still obstacles set by the CAD system vendors who have developed CAD system competitively and separately. STEP standards define conformance testing methodology and framework (ISO 1994d) for implementors to comply with, also define some conformance requirements in particular application protocols, such as the six conformance classes in STEP 203 and the protocol information and conformance statement (PICS) proforma that supports conformance assessment (ISO 1994f). Besides, many programming tools and APIs are available for supporting standard compliance, such as STEP Tools (STEP Tools Inc. 2012). However, these facilities have limited enforcement over application vendors. The level of complying with the STEP standard and how to implement the detail are still largely in the hands of CAD vendors. For example, the conformance class for geometric representation can be complied with, but how to process some generic attributes may vary. According to our experimental results, both Autodesk Inventor and CATIA support importing STEP files and can pick up the labels at body level but not other entities, such as face and edges. This cultural factor is obviously out of the control of end users.

However, as an earlier experiment suggests, there can be resolvable issues, such as transferring certain arttributes programmatically (e.g. labels), where two ways were identified. The first one can be the experimental approach described in Section 4.1.4. This is to compare the definitions of each interested entities defined in the intermediate STEP file against the entities rendered in a target CAD system, for instance, the label ‘FACE_1’ of entity ‘ADVANCED_FACE’ in Table 48. The other one may use the same comparison algorithm, but instead uses the specifications semantically stored in the OKB rather than the original STEP file as illustrated in Table 48. The latter approach can be achieved by applying a full complied STEP model in the ontologies, which is topologically self-contained and independent from conventional geometric formats. However, this requires a fully transformation from STEP into an ontological representation, which is not implemented in the prototype.

Other than consistently mapping anchors for a static model across CAD systems, there is another criterion in persistency: to identify modified elements of the original model in the modified model. Mun and Han (2005) proposed mechanisms to solve the ambiguity issues for entity splitting and merging based on historical data of a CAD model. There are many other works in this paradigm, including the works of Marcheix and Pierra (2002),

Agbodan et al. (2003), Bidarra et al. (2005); and Baba-Ali et al. (2009). However, further exploration in this issue can be seen as a separate research work which is beyond the scope of this presented work. We focus more on providing a general information platform that has persistent (at least consistent) anchors associated with CAD systems.

Table 48 Comparison between Examples of a Face in STEP and Ontological Representations

STEP File
#192=ADVANCED_FACE ('FACE_1',(#281),#255,.F.); *…… omitted ……* #255=CYLINDRICAL_SURFACE('',#1183,8.5); *…… omitted ……* #1183=AXIS2_PLACEMENT_3D('',#1618,#1313,#1314); *…… omitted ……* #1313=DIRECTION ('',(0.,1.,0.)); #1314=DIRECTION ('',(0.,0.,1.)); *…… omitted ……* #1618=CARTESIAN_POINT ('',(45.,6.,0.));
Ontological Representation
BODY_1 hasAdvancedFace FACE_1 FACE_1 hasCylindricalSurface CYLINDRICAL_SURFACE_1 CYLINDRICAL_SURFACE_1 hasAxis2Placement3D AXIS2_PLACEMENT_3D_1 *…… omitted ……* AXIS2_PLACEMENT_3D_1 hasDirectionRatios 0 AXIS2_PLACEMENT_3D_1 hasDirectionRatios 1 AXIS2_PLACEMENT_3D_1 hasDirectionRatios 0 *…… omitted ……* CYLINDRICAL_SURFACE_1 hasCartesionPoint CARTESIAN_POINT_1 CARTESIAN_POINT_1 hasCoordinate 45 CARTESIAN_POINT_1 hasCoordinate 6 CARTESIAN_POINT_1 hasCoordinate 0 *…… omitted ……*

8.1.2 The Coverage of Information Objects and Anchoring Granularities

During design collaboration, there is a need for accurate and explicit interpretation of design intent among teams or designers (Pahl et al. 2007). Direct reference to all levels of granularity in a design object offers such ability, which is the role of annotation anchors. Davies' work (2008) envisages covering all levels of granularity but only has designed and implemented at the level of modelling features, which is not formally and systematically defined. Therefore self-contained semantic anchoring in his approach is practically difficult. In another earlier work, a knowledge intensive engineering framework (KIEF) from (Yoshioka et al. 2004) uses generic building blocks (features) as anchors, on which FBS knowledge base are built. This is implemented for assemblies and maybe feature-based modelling geometry models, but not B-rep models. Thus direct references

to further detailed geometric entities are not possible in KIEF. As a result, some engineering analysis would not achieve high accuracy, such as parametric or detailed cost estimation and FEA. Similarly, in the work at Washington State University (Zhu et al. 2009), knowledge bases are built at the assembly and part level, and loosely joined with CAD systems.

In contrast, the proposed framework in this thesis uses a standard-compliant semantic mechanism to describe B-rep geometric models. This enables knowledge to be associated with all levels of granularity persistently (or consistently at least), including assembly, body, face, edge and vertex. This opens the possibility for all kind of knowledge to be directly associated with specific levels of granularity. This feature provides the foundation for fusing the CAD system and the knowledge base while still offering a stand-off ability to allow users to have full access control over the knowledge base.

However, one obstacle that has been identified during the experimental work is the computing resources required to reason over large OWL ontologies. It takes unacceptable computing time for a reasoner to process hundreds of instances and classes on every change of the ontologies, e.g. vary from few seconds up to hours depending on the complexity of ontologies. This seriously hinders an immediate application of the proposed approach. For this reason, vertex level is not yet designed and implemented in the prototype, but only to demonstrate the potential to be extended. Fortunately, the efficiency of OWL reasoners has been notably improved continually according to the observation of the author during the past few years. And also the large supporting community keeps ontological technologies promising and fertile.

8.1.3 The Coverage of Target Media Types

Not only can the coverage of granularities on CAD models be extended, the coverage of target media types is also extendable using the same metadata mechanism proposed in this work. The attempt of adding free-style comments described in Section 6.3.1 and Section 7.3.1 implies that this semantic anchoring mechanism can be applied to other target media referenced by an assigned URI, such as digital text documents, multimedia documents and so on. Based on this assumption, this proposed OntoCAD system may be extended to cover all known digital documents and at various levels of granularity.

Take text documents as example, a text file can be readily referenced by an anchor with a data value containing an URI pointing to the file. In the work of Wang (2005), a persistent annotation scheme aimed at text documents has been proposed. Three types of location

descriptors are proposed to define annotation anchors in order to address anchoring persistency over dynamic text documents: meta-structure information location descriptors, the keyword location descriptors, and the semantic concept location descriptors. In consideration of the case of OntoCAD, these location descriptors can be potentially conceptualized and instantiated in the OKB, and thus extend the coverage to include text documents.

Similarly, techniques such as atoms and clumps (Ovsiannikov et al. 1999) have been proposed to represent the smallest text unit (e.g. a word) or a freehand or square selection of a picture and used as anchor descriptors. Again, static raw data (Alink 2005), multimedia documents (Schroeter et al. 2006) including images, audio and video contents, and also 3D objects can be used to identify the region of digital objects as anchors. All these metadata approaches show the potential to be incorporated into the OntoCAD framework.

With this prospect, OntoCAD is currently ideal for but not limited to CAD parts. It can be extended to cover various target media types, which improves the evolvability of such a system. As a consequence, OntoCAD is not limited to the application at late design stages after CAD systems are involved. By incorporating other design documents such as design requirement and specification documents, analysis reports and so on, the OntoCAD framework can encompass the entire design process. Furthermore, with other desirable features such as distributed collaboration (Wang and Nnaji 2006; Yu et al. 2010), lightweight product data representation (Ding et al. 2009), other documents in manufacturing, services, until recycling/disposal can be incorporated too. Therefore, OntoCAD can be seen as a general framework that can potentially provide a closed loop of knowledge and information throughout the PLC.

8.2 Annotation Content

In the previous section, how annotation anchors are specified has been discussed. This section continues with the discussion on the other critical component – annotation content. This includes the issues on how to represent, store, retrieve and reuse knowledge, and how to control the association of knowledge element with anchors. The difference from other researchers' work, and the advantages, the difficulties and limitations of this proposed work will also be discussed.

8.2.1 Knowledge Acquisition and Representation

Based on the successful demonstration of applicability and feasibility of a prototype system in two use cases – cost estimation and FEA, it can be concluded that this proposed framework has set a new foundation for knowledge and information use with CAD models. This stand-off structure allows knowledge base to be developed, maintained and even used independently from the CAD environment, namely it can be operated through ontology editors.

This is a fundamental difference from Davies' work (2008), in which the annotation data is structured as user defined objects (UDOs) – a native feature of a certain CAD system (i.e. Siemens' NX). This feature allows users to add their own custom object inside of NX for particular purposes. However, the data is stored in the UDO and the behaviour has to be defined in NX programmatically. From this point of view, it is a more tool-oriented data structure, rather than a knowledge-object-oriented data structure as proposed in the OntoCAD. The ontological classes defined in the OKB are the corresponding data structure in Davies' work, and engineering constraints in multiple viewpoints are defined as behaviours on top of this data structure. Although some functions are still realized through CAD systems, such as information capture through the OGUI, and query and reasoning by the OMA, knowledge is largely self-contained in the ontologies.

Furthermore, this proposed work is relatively more consolidated in terms of a formal syntax, schema and a prototype system, which is a step further from Davies' work. Although automation processes for MEV engineering evaluation have been described and partially demonstrated in Davies' work, a formal language consisting of syntax to specify the metadata and a schema to represent knowledge are not explicitly defined. This is the great obstacle to knowledge process automation, especially through reasoning over a self-contained knowledge base. With regard to demonstration, there is not an integral prototype system developed eventually to systematically evaluate his concept.

The OKB in the OntoCAD is composed of conceptually modularised ontologies. Firstly, it enables knowledge to be represented hierarchically and compartmentalised with roles. For examples, the FO is associated with fundamental knowledge and geometric anchors; the EVOs represent the general viewpoint-dependent knowledge; and the AOs are in relation to vocabularies of specific tools. This structure is originally inspired by the FBS structure (Umeda et al. 1990; Qian and Gero 1996; Gero and Kannengiesser 2004; Colombo et al. 2007), and has great parallel with the knowledge structure proposed in KIEF (Yoshioka et al. 2004) and the work of Zhu et al. (2009). The most significant

difference from their work is the broader coverage of anchors and the tighter association with all types of target media, which has been discussed in the previous section. More importantly, there is foundamental difference in knowledge base architecture. Terms for CAE applications including geometric information in the work of Zhu et al, is in the scope of application specific ontologies. In this presented work, an object-oriented flavour is adopted. In other words, geometric related information is used as anchors, belongs to the scope of foundamental ontologies. Thus all other knowledge and information root from it. And this root is extendable to cover other types of media. This is more intuitive and tracable to knowledge modellers.

With regard to knowledge acquisition, the tighter association between CAD models and the knowledge base not only refers to the anchors, but also the data population from the CAD model into the knowledge base. In manual knowledge acquisition, OntoCAD has realised a mechanism for an adaptable intuitive user interface that Davies envisaged. OntoCAD allows users not only add the information but begin to design how they add it by configuring the background knowledge base. For example, which EVO has association with the geometric entity type – face and also its contents can be constrained. In the automatic knowledge acquisition process, OntoCAD considers both pre-population and real-time population processes for CAD related data (Section 6.3.3). The pre-population can improve the process automation for data collection and also avoid tedious and error-prone human operations, while the real-time population is able to handle modification of CAD models. In comparison, the work of Kiryakov et al. (2004), only considered pre-population processes. This may be suitable for static objects, but will neglect changes in dynamic objects. The OntoCAD approach aims at the later situation as CAD models may be modified during design process, including geometric information and also others, such as tolerances, dimensions, material and manufacturing.

Secondly, the modularity separates the roles of fundamental use, general engineering viewpoint dependent use and specific application use. The fundamental use refers to the semantic self-awareness of topology. Based on the capability of OWL ontologies, this has been discussed to be theoretically feasible however not implemented in this research work. Based on EVOs, engineering constraints can be defined and operated, therefore to provide general viewpoint-based services, such as giving advice based on costing design or manufacturing design as described in Section 5.5 and Section 7.2.7. And the specific application use can be supported by the cooperation between AOs and AWs as described in Section 7.2.5.

Moreover, this modularity gives a full access control to the owner of the design object. In

an ideal support, ontological modules can be imported as needed and the unnecessary modules can be removed to avoid disturbance or exposing sensitive intellectual property (IP) (Ding et al. 2009).

Owing to the constrained research scope defined at the beginning and the time constraint, this OntoCAD framework proposed in this thesis has some limitations. The actual support has been developed relatively coarsely to demonstrate the core contribution only. It can certainly be refined to achieve more complete knowledge modules. For example, the incomplete of the levels of granularity such as assembly, vertexes and customised features, and the associated concepts have not been implemented. As a consequence, parameters related to these types of entities in cost estimation and FEA can not be demonstrated yet, such as identifying topologically crossed holes, annotate a load condition on a point and so on.

Moreover, the concept of knowledge base modularity has been raised but not explored further and not fully implemented in this present work. In the experimental work, ontologies were conceptually partitioned into hierarchies but physically stored as a whole. It was attempted to separate all EVOs and AOs from the FO, and use these knowledge modules in a plug-in flavour. Unfortunately, the implementation work revealed it is almost impossible to totally separate ontologies, especially for EVOs. This is due to the complex interrelations among them.

It is even more difficult to automatically import an external ontology. In our experimental work, an attempt was made to import an external ontology for unit systems (for weight, volume etc.) using the Protégé ‘import’ capability and to merge it with the OKB. However, the Protégé editing tool can only blindly import ontologies without intelligent knowledge mapping. This caused duplicated classes and confusion. Although the Protégé tool provides a merging function, it is not yet sophisticated enough to achieve 100% accuracy. This is mainly due to the insufficient knowledge-level mapping between the being imported ontology and the existing ontology. It is still an open issue, and seems unrealistic for any available tools just yet (Falconer et al. 2007).

The automatic knowledge mapping has been a popular but unsolved subject in the ontology research field. Many approaches have been proposed, such as heuristics method (Zhan 2007; Kim et al. 2009), parametric approach (Pirró and Talia 2010), and an adaptive approach (Mao et al. 2010). This can be a future work but beyond the scope of current research. Any integration with external knowledge modules is a manual process by following the proposed OntoCAD knowledge modelling methodology as guidelines.

8.2.2 Knowledge Sharing and Use

It has been successfully demonstrated that knowledge can be shared based on an assumption that manufacturing process knowledge is generated from one CAE tool, which can be used to serve cost estimation in another. Other information such as material, data types, measurement units and so on were also demonstrated as shareable. This therefore deduces that knowledge as fundamental concepts, engineering viewpoints or specific application/service can be reused either internally or exported to an external application.

This feature has been enabled and advanced by the reasoning service. Rather than looking at improving the inference algorithm, expressiveness of description logic or decidability (Horrocks 2005; Eiter et al. 2006; Fensel et al. 2008), it has been stressed on how to use the standard reasoning service to aid in engineering tasks (Section 5.5 and Section 7.2.7). As discussed in Section 8.2.1, two types of engineering tasks depending on all three types of reasoning (Section 5.5) have been described. One is based on EVOs, such as giving viewpoint-based advice, or further automatic actions that can be taken according to the advice; the other one is based on AOs, such as providing dataset for external tools thus to help tool integration. In addition, the conceptual reasoning provides concept mapping through equivalency checking. These features provide procedural ability and higher level of process automation in comparing with the KIEF approach by (Yoshioka et al. 2004).

In a similar way to many other research works including (Anthony et al. 2001; Kitamura et al. 2006; Colombo et al. 2007), they are all more or less in the paradigm of FBS. However, most of them stress how to describe functions and behaviours at the high level of structure, often at least the level of design part. This decides that the engineering evaluation can not be processed in detailed way as by OntoCAD, which makes many parameter-based automatic engineering evaluations impossible (e.g. parametric cost estimation).

According to the literature research, it has been confirmed that the semantic expressiveness can be significantly improved with the extension languages such as SWRL and SQWRL. For example, the build-ins of SWRL can describe mathematical relations, therefore some procedural operations may be setup within the knowledge base rather than programmed in an external environment, such as unit conversion and other equations based on physical laws (Horrocks et al. 2004). For another example, the collection operators of SQWRL may use closure operations to complete the weakness of

the open world assumption in some situations (Stanford Medical Informatics 2010).

Although these extensions are in principle supported, they were not included and implemented into OntoCAD. This was due to unavailability of such features in the version of the ontology editor (i.e. 4.2 of Protégé) used during the experimental work. SWRL support was only available for versions some 10 releases before the current version (the last one was version 3.4.8), and with outdated version of OWL. This meant that defining classes with SWRL rules and further processes in the experimental work would have been very difficult and tedious.

8.3 Knowledge Modelling Methodology

In the domain of CAE, 81% of the reviewed research work (37 papers) in the development of KBS did not comply with any knowledge modelling methodology as claimed by Verhagen et al. (2011). According to their critical review, many KBS were developed using an ad-hoc process based on a particular case rather than adhering to a structural framework or a formal methodology. For instance, the ontology presented in an early work PACT (Cutkosky et al. 1993) was informally documented in e-mail messages, rather than following a systematic methodology. Formal guidelines in the more recent work such as (Zhu et al. 2009; Matsokis and Kiritsis 2010) are still missing. The absence of a systematic methodology may result in lower generality of an approach. The proposed OntoCAD framework addressed this issue by a hybrid of two strategies of knowledge modelling methodology, which can be applied to suit different situations, either to develop FO or EVO from scratch or to develop an AO.

The evaluation based on two cases – cost estimation and FEA – has revealed that it is more guided and easier to commence modelling an EVO or AO using a bottom-up strategy where appropriate, and it probably takes a longer time if using a middle-out strategy. This is due to that the bottom-up strategy is more straightforward and better guided with explicit literature support, such as handbooks of a specific application, user manuals etc. Furthermore, the bottom-up strategy is less affected by the participants other than the knowledge modeller. In comparison, time consumption and uncertainty in the early modelling stages before codification have been affected by the human factors in the middle-out strategy. This has been revealed in the experimental work, such as the quality of questionnaire design, the availability of interviewees, and feedback validity. For example, the questionnaire for the same type of expertise may be understood and interpreted differently by the interviewees with minor differences in background, e.g. cost experts who have strong opinion on manufacturing process, or service costing.

In a nutshell, it is difficult to conclude which strategy is more efficient or advanced. It is more dependent on the situation of the specific case and the complexity of the ontology being developed. But it can be concluded that bottom-up is preferred over middle-out strategy wherever it is appropriate, in order to benefit from a more guided development lifecycle and maybe a shorter one too. This conclusion is based on the assumption that it is relatively comparable for the complexity of the questioned EVs or applications and learning curve required. The learning curve caused by adapting to the methodology itself has not been considered.

Even though many methodologies have been developed overtime and some practical guidelines have also been available, it has been found by Verhagen et al. (2011) that methodology adherence can not be enforced due to the unavailability of supporting tools and technologies.

8.4 The Generality of OntoCAD

The proposed OntoCAD framework aims to provide a general-purpose knowledge and information management system to aid engineering design processes through multiple engineering viewpoint knowledge sharing and reuse. Through the prototype system and two use cases, it has been evaluated and concluded that it has met the defined requirements, which reflects the success in the means of feasibility and applicability of the annotation scheme and also the methodology.

Although two cases are far from enough, it can be assumed that other engineering applications or services can be readily incorporated by extending the OKB with new/updated EVOs or/and AOs, however with few or even without programming changes in the application. For example, based on the EVO_Material, EVO_Cost and other EVOs, EVO_Manufacturing can be expanded. Also AWs can be defined with engineering constraints, such as those reflect the applicability of manufacturing processes. In this way, annotations can be made to suggest desired manufacturing processes in considering material properties and manufacturing costs or giving general advice.

Furthermore, this OntoCAD framework has been accordant with a research tendency in KBS, which is the separation between knowledge and implementation of software development (Verhagen et al. 2011). Two research issues need to be noted in this separation. One is to improve the transparency and accessibility to the knowledge base of a KIM system. This means the direct access though knowledge management applications, such as the Protégé ontology editor in our case. This offers users opportunity to move

away from black-box applications and to customise the knowledge base. The other one is to enrich semantics in knowledge models. This is about improving data interoperability. It is also about knowledge transformation from tacit towards explicit, so that some programmed functions can be moved into the more application-dependent knowledge base. This has been demonstrated by the procedural annotation feature realized with AWs. This separation theoretically improves the generality of KBSs.

8.5 Future Work

Elicited from these discussions, some potential directions for future work can be derived: completion of levels of granularity and more advanced semantics in the annotation anchoring mechanism, knowledge base management and maintenance (modularity and mapping), advanced reasoning for higher levels of knowledge process automation, level of formality (synergy of formal knowledge and informal knowledge), wider applicability in the engineering domain and interface to other fields.

8.5.1 Anchoring Mechanism

From the implementation viewpoint, the completion of levels of anchoring granularity on CAD models is desired. For example, assemblies, vertexes, geometric modelling features and so on can be included as discussed. This will enable the OntoCAD to annotate BOM, more sound FEA models, or other engineering viewpoints. Broader coverage of target object media types is also desired to improve the collaboration environment, such as to support digital text document, annotating images and multimedia documents.

More advanced semantics in annotation anchoring mechanisms could be another research direction. This means to enrich the semantics lying among geometric elements, so that the elements can understand themselves including topology. This may potentially aid feature recognition and persistent anchoring. Multidirectional anchors may also be a way to improve the traceability of information. For example as well as inquiring annotation contents based on given anchors, anchors could be inquired upon based on given annotation contents, or all concerned anchors and further annotation entries searched by giving a piece of intermediate information.

8.5.2 Degree Of Formality

The formality degree of knowledge representation has been a difficult decision. Higher degree of formality certainly enables reasoning possibility and improves the process-ability but may decrease the expressiveness. In contrast, higher expressiveness

may easily cope with complex situation, but may compromise the process-ability by computer agent (Webster 1988; Gašević et al. 2006a; Stephan et al. 2007). For example, a freestyle annotation “This part is manufactured using a sand casting process” can be easily expressed; however an equivalent formal annotation “BODY_1 hasManufacturingProcess Sand_Casting_01” can be more explicitly processed. The present work has been mainly focused on representing engineering expertise as formal knowledge. Some research directly processes informal knowledge using techniques such as NLP (Setchi et al. 2011). From the author’s viewpoint, the synergy of formal knowledge and informal knowledge can be an improvement for higher level of automation in knowledge process, namely, allowing users to input information intuitively, and the information can be automatically parsed at background and update to the formal knowledge base.

8.5.3 Advanced Reasoning

Although knowledge sharing and reuse have been successfully demonstrated, there are many more potentials that have not been explored. As noted, SWRL, SQWRL and their build-ins have potential to further improve the expression of logics and mathematical relations. With such expressions, semantics can be enriched and more advanced reasoning actions enabled, and a higher level of knowledge process automation or process reuse can be expected.

Furthermore, machine learning technologies can be considered to improve knowledge reuse, namely to improve the ability of the reasoner/ontology to learn from itself based on the behaviours or experiences (Setchi and Tang 2007). Therefore the system in terms of agent organization can be upgraded from broker to mediator (Shen et al. 2006). Ultimately, the system can be upgraded from a federation approach to an autonomous agent approach, so that the system entirely manages itself.

8.5.4 Knowledge Base Management And Maintenance

The concept of knowledge base modularity has been raised but not further explored in this present work, which is worth doing. This includes ontology exchange (i.e. exportation and importation), separation of Abox and Tbox for maintenance and computing efficiency, and knowledge mapping technologies.

As discussed, the knowledge modelling methodology has a positive effect to knowledge management, which in turn supports evolvability and higher generality of the total system. However, it lacks support in methodology enforcement (Verhagen et al. 2011). Tools and

technologies are needed to constrain modellers' behaviour and development process, to aid in automatic knowledge codification thus to reduce human mistakes

There are certainly many other research directions ongoing (Verhagen et al. 2011), such as supporting distributed network environment (Shen et al. 2006; Yu et al. 2010), access control and security (Zhou et al. 2008; Bertram et al. 2010), lifecycle management for 4D design models (time as fourth dimension) (Mahalingam et al. 2010) and so on. These features could be incoporated into the framework presented in this thesis, therefore to enable it to aid in collaboration further.

Chapter 9 Conclusion

In this PhD research project, an initial exploration in knowledge and information management is carried out, and important research challenges in the mechanical engineering domain are identified: knowledge management especially knowledge acquisition, representation and maintenance, data interoperability, data and tool integration, and many others. This is even more critical when a detailed digital design objects become available through CAE systems. In order to address these research challenges, an overall aim is conceptualized, which is set to explore and define a systematic semantic annotation framework to assist with CAD-based design in the domain of mechanical engineering at late design stages. To achieve this overall aim a list of objectives has been raised in Section 1.1:

Objective 1 To explore the literature related to PLM, engineering design, CAD, MEV and then the related technologies and the applications in various domains, with an emphasis on mechanical engineering. Thus to identify research gaps in engineering knowledge management.

Objective 2 To define an extendable framework to systematically manage knowledge in supporting MEV in order to assist with knowledge acquisition (KA), knowledge representation, data and tool integration and interoperability.

Objective 3 To provide a data structure that is able to accommodate knowledge and allow it to be associated with design objects.

Objective 4 To define a mechanism that supports the query and exchange of data, and to derive new knowledge based on existing knowledge.

Objective 5 To provide guidelines for a rigorous methodology to assist with knowledge modelling to allow the proposed framework to be maintainable and extendable.

Objective 6 To design and develop a demonstration system, and thus to evaluate the success or otherwise of the proposed framework in terms of the feasibility, effectiveness and generality.

In order to guide the research and encourage reflection during the research lifetime, the design research methodology (DRM) introduced by Blessing and Chakrabarti (2009) Rather than repeating these four DRM stages, this final chapter concludes the presented work according to the aim and objectives, from the initial research for identifying research challenges, a solution to improve current situation, how this proposal has been

implemented and evaluated, to what have been learnt. And then the key achievement and contributions are concluded at the end of this chapter.

9.1 Objective 1 – Research Clarification

The presented work starts with background research. Product lifecycle management (PLM) is firstly described as a series of processes that manage the entire product lifecycle (PLC) including portfolio management, product design, process design, supply, production, product launch, service and support, end of life and recycling. Its computational enabler – the PLM systems have been reviewed, including their history and the state of the art of commercial applications. It has been found that resource requirements for deploying PLM systems, data integration, design collaboration and knowledge processing are the main challenges.

CAD systems have been developed and used as a common practice for modelling and communicating to aid productivity and automation in engineering design process, especially at the late design stages: embodiment design and detail design. To some extent, CAD systems either interact with PLM systems or have been integrated as an important part of PLM environment. The history and the state of the art of CAD system in both commercial applications and academic research have been explored. As a results, it has been found that there are still research gaps, including automatic knowledge processing, knowledge management especially knowledge representation, collaboration among users and across different CAD systems, data interoperability and integrated comprehensive knowledge model or simplified knowledge models to aid different purposes. As these findings imply, knowledge management is identified as a critical factor for improvement.

Therefore, knowledge management is also reviewed, in which the processes include knowledge acquisition and representation, storing, retrieval and use. Capturing and representing knowledge are still open issues in order to transform tacit knowledge into explicit knowledge. In regarding to aid engineering design, the concept of multiple engineering viewpoints (MEV) can be seen as a support to assist with knowledge management. MEV refers to multifaceted concept with multilayered meaning from multidiscipline or viewpoints. The function-behaviour-structure (/state) (FBS) knowledge model and the derived multilayered knowledge models are some important forms of support.

To help understanding the concept of EV and some particular problems in engineering

design, two cases of engineering analysis were described: cost estimation and finite element analysis (FEA). Each case is an EV that requires the corresponding expertise, in which knowledge from other EVs are also involved. In the first case, an ideal situation is to integrate an external cost analysis tool into a CAD system so that cost estimation becomes an embedded service to provide real-time decision making support. In the second case, it will be useful if the legacy service – finite element analysis related knowledge - can be recorded as a general model that can be shared by other CAD systems or other tools/services.

9.2 Objective 1 – Further Review on Computational Enablers

In order to improve knowledge and information management, two computational enablers and their related technologies have been reviewed: annotation and ontology.

In digital context, annotation is extra information added to target object, and consists of two components: annotation anchor and annotation content. Annotations have been used ubiquitously. Over thirty existing annotation approaches and their applications were reviewed, and classified in terms of six dimensions: the targeted media, the audience, the rendering system, the usage and function, the representation and the storage location. Based on these proposed classifications, some reviewed approaches are mapped into a table for comparison. It has been found that the more complex and advanced the annotation approach is, the more categories it may fall into. Most approaches are not mutually exclusive in these classifications; in fact, many of them have multiple features.

According to the explored literature and a preliminary experimental work, it can be concluded that conventional annotation can be used to capture and represent data while maintain association between additional information and target. However, there are still challenges in order to address knowledge management based on CAD systems, including robustness and granularity of annotation anchors, data structures for knowledge representation rather than data/information only and mechanisms for further management. An important trend in annotation technologies is semantic annotation that tries to address these weaknesses, which has been intensively explored in Semantic Web but deficiently studied in mechanical engineering. Without semantics, annotations are mainly a media to represent data, rather than knowledge. Therefore, as an important computational enabler to knowledge representation, ontological technologies have been studied.

Ontology is a set of knowledge terms that consist of the vocabulary, the semantic interconnections, rules of inference and logic for some particular topic in an interested

domain. As a formal knowledge representation, ontology needs to be specified with a formal specification language with an editing tool. Considering language capability, the complementary rule and query languages, and the popularity, the web ontology language (OWL) is concluded as the most suitable language. There are also supporting languages to OWL, which strengthen its expressiveness, such as semantic web rule language (SWRL) and semantic query-enhanced web rule language (SQWRL). And in consideration of open source, maturity and large active user community, Protégé is concluded as a suitable tool for ontology modelling. Furthermore, to support ontology modelling, methodologies are also reviewed, which mainly have three strategies: top-down, middle-out and bottom-up.

The state of the art in ontology development and applications has also been reviewed, especially in engineering domain. Some research issues were identified, mainly in three aspects: an efficient interface to knowledge acquisition based on CAD systems during the late engineering design stages and allowing intuitive user intervention; ontology architectures that are capable of incorporating MEV to assist with engineering knowledge management; the unsolved system integration issue. Furthermore, although a key feature of ontologies is reasoning, advanced reasoning mechanisms and the applications in engineering domain is still under-developed.

In a nutshell, annotation holds the potential to provide external interface and basic data structure, while ontology holds the potential to represent, manage, query, retrieve and reuse knowledge. However, both of them have their weaknesses if they work alone, and the mechanism to combine their merits is still missing. As a fact, an integral system based on CAD systems to provide a closed loop of knowledge and information is not yet available.

9.3 Objective 2 and 3 – An Extendable Knowledge-Based Framework

Based on the findings in Objective 1, one of the most important factors need to be improved is knowledge management. This is mainly reflecting to the Objective 2, in which an extendable framework is needed to systematically manage knowledge. It requires the capability to incorporate MEV in order to assist with all knowledge management processes, including knowledge acquisition (KA), knowledge representation (KR), and how to use the knowledge to serve data and tool integration and interoperability. The other important factor is data. If a knowledge base can be provided by achieving Objective 2, the next step would be define a formal data structure to accommodate knowledge and populate data into it, and more importantly, how the knowledge and data

can be coherently associated with the design objects. The design objects can be text documents, forms, multimedia documents, meeting minutes and so on. In this PhD research work, the most primary design objects are CAD models. This second step mainly reflects to the Objective 3.

Rather than deal with Objective 2 (i.e. the knowledge architecture), Objective 3 is defined first since it is the infrastructure – a prerequisite to everything else. A novel annotation data structure has been proposed, which seamlessly couples the knowledge and data. The conventional annotation structure is adapted in this solution: annotation anchor and annotation content. It is mainly aimed to solve the issue of association between the forthcoming knowledge base and the design objects. Since OntoCAD will be an approach to improve the automation of knowledge processes, the metadata needs a formal specification scheme. Hence, OWL is employed as a coherent knowledge and information specification language to describe the metadata for both annotation anchor and annotation content. This is achieved by using the OWL object properties (a linkage of objects) and the OWL data properties (a linkage of object and data), thus object annotations and data annotations can be constructed respectively. Based on these two basic types, an annotation chain can be formed with direct annotation node and the rest indirect annotation nodes. These two types of nodes differ in whether associate with the primary design objects – CAD models. Hence, a new semantic metadata structure is defined particularly in aid of annotations.

Since annotations are anchored on CAD models, the geometric models and their constituents need to be represented in the knowledge base. With regard to data interoperability, this fundamental annotation data structure has been further strengthened by complying with an industrial standard family – **ST**andard for the **E**xchange of **P**roduct model data (STEP). This standard family covers a broad spectrum, such as geometric representation, data encoding, specification languages and application protocols. By conforming to STEP, it refers to using its data types, schema, and boundary representation (B-rep) and so on as appropriate, but the metadata is specified in OWL. In other words, it is to translate essential parts of a CAD model from STEP into OWL. Based on this improvement, a consensual geometric model and other data are established and can be extended upon. Hence the persistency of annotation anchoring can also be improved when transplanting a knowledge base across CAD systems.

According to this infrastructural data structure, a systematic architecture for a knowledge base has been proposed to address Objective 2 for higher level of knowledge abstract. This knowledge base adopted a three-layered architecture that is composed of

Foundation Ontology (FO), Engineering Viewpoint Ontologies (EVOs) and Application Ontologies (AOs). The aforementioned STEP-compliant geometric representations, fundamental datatypes, unit systems and other CAD or project related general knowledge fall in the scope of the top layer – FO. The second layer is a collection of ontologies representing engineering viewpoints (EV), such as manufacturing, material, cost estimation, FEA and so on. These EVOs may overlap or reference each other, and all of them are associated with the FO. The bottom layer is a collection of ontologies representing terms in regard to specific applications, such as a vocabulary for a costing tool SEER-MFG. AOs are associated with corresponding EVOs or directly with the FO. Therefore all knowledge elements are associated as a network to propagate knowledge across layers in the knowledge base.

Based upon this data structure and knowledge base architecture, a knowledge-based system (KBS) is realized to address Objective 2 and Objective 3 in an integral way. The OntoCAD (Ontology-driven semantic annotation framework for CAD systems) is therefore proposed as a novel and general KBS to aid in the overall aim, namely to finally joint the ontological knowledge base with a CAD system, thus to capture, represent, manage and use MEV knowledge in aid of engineering design processes at late design stages. The OntoCAD system consists of three key modules: OntoCAD Graphical User Interface (OGUI), OntoCAD Knowledge Base (OKB) and OntoCAD MEV Agent (OMA). The OGUI is an interface embedded in a CAD system for end users to capture inputs. The OKB is in fact the aforementioned layered knowledge base containing expertise as multiple engineering viewpoints. The OMA is an intelligent broker that aids the synergy among CAD system (i.e. data collection), the knowledge base (i.e. knowledge population and query) and being integrated external engineering tools/services (i.e. knowledge and information exchange). This is the first general semantic annotation approach known to the author that builds on ontological knowledge base and tightly couple with CAD systems with full manipulation granularities.

9.4 Objective 4 – To Use the Knowledge

Objective 2 and Objective 3 have been achieved by this OntoCAD system. To take the most advantage of this system, mechanisms are also suggested to address Objective 4, namely for queries, data exchange and deriving new knowledge based on existing knowledge.

The standard query can be achieved by retrieving the classes or data value of an asserted OWL entity. For examples, by giving an individual, its classes (types) or

associated data can be retrieved. This forms the basis for data exchange. To achieve automation of data exchange, it needs the help of intelligent computer agents, which is the role played by the OMA. For examples, the OMA together with OGUI provide interfaces for user to feed in an input file containing a set of query parameters, and then process automatic queries to satisfy the required dataset.

Apart from these basic features, reasoning services provided by the OMA are capable to infer new knowledge over the existing knowledge base. The reasoning services have been classified into three categories: factual reasoning at data level, conceptual reasoning at class level and methodological reasoning for the combination of data and concepts. The factual reasoning computes individual membership, and contributes to application watchdogs (AW) in the methodological reasoning. The conceptual reasoning computes the class subsumption including class equivalency (knowledge mapping). It is mainly realized in providing an intuitive user interfaces and build associations between ontological concepts. Both of these two standard reasoning services can be used for consistency checking, which ensures no contradiction exists in the ontology model.

The methodological reasoning is a combination reasoning based on the standard reasoning services. An important application is the AWs. An AW is an assigned agent to monitor the states of satisfaction on a set of queries for a tool/service, or it watches whether an engineering rule is complied by a certain type of entities. If a type of an geometric element has association with all queries in a dataset, it will be selected as candidate class including its equivalent classes (conceptual reasoning); if any individual of these candidate classes satisfies the constraints on data (individual membership), this individual will be highlighted by this particular AW and ready for downstream processes. The down stream processes may be to automatically export the dataset in a required format. In this way, OntoCAD gains the ability to integrate tools/services along more EVOs and AOs are incorporated. Apart from integrating downstream tools, the OntoCAD may give general advice based on an engineering rule for users' consideration by reasoning over the FO and EVOS.

In a nutshell, the Objective 4 is accomplished with the features of basic data retrieval and the advanced semantic data query. The novelty here is to use reasoning facilities over an ontological knowledge base to serve engineering tasks. This covers not only general engineering tasks, but also tool specific tasks. Therefore, it opens the possibility to seamless integrate legacy or new engineering applications, and improve process automation.

9.5 Objective 5 – To Develop and Maintain the Knowledge

Since this presented OntoCAD system is driven by the modularized ontological knowledge base, the efficiency and applicability are largely affected by how well the development and maintenance of this knowledge base, i.e. Objective 5. To regulate and smooth such processes, a general OntoCAD ontology modelling methodology has been introduced.

The proposed OntoCAD methodology combines two strategies to suit different situations: middle-out and bottom-up. In the case of modelling FO from scratch or adding new EVO for general use without a particular downstream application, a middle-out route can be adopted. On the other hand, if it is about integrating a particular application into existing ontologies rather than from scratch, a bottom-up route can be followed. The bottom-up strategy joins the middle-out approach in the late common stages, including implementation, evaluation and documentation. This study contributes to completing this presented framework more sound, mature and formally maintainable.

9.6 Objective 6 – A OntoCAD Prototype System and Evaluation

With regard to Objective 6, an actual support – OntoCAD prototype system has been designed, developed and evaluated to demonstrate the feasibility, effectiveness and generality of this proposed framework.

This prototype operates as an add-on application to a commercial CAD system NX6. The completeness of the knowledge base, user experience and the level of sophistication of the software application have not been the primary goal in the prototype design. Instead, the core functionalities have been specially paid attention and implemented with some necessary and reasonable compromised manual setup due to the time constraint of this project. The data structure has strictly followed the OntoCAD annotation data structure. The prototype has also adopted the presented three-module architecture (i.e. OGUI, OKB and OMA), in which the OKB has the three-layered MEV ontologies. The application has been programmed in JAVA language, with the employment of some APIs, including NX OPEN API for the OGUI and accessing the CAD models, OWL API and Pellet for accessing the OKB, and a spreadsheet API for data exchange.

During this research work, evaluation has been planned and operated as an essential process for verifying and validating this proposed framework and the assumptions it based on. Three aspects have been addressed in the evaluation processes: application evaluation, support evaluation and success evaluation. The application evaluation

validated the applicability and usability of the proposed support against the desired key factors. It has been done during the background research and the preliminary experimental work. During the design and development of this prototype, the support evaluation (i.e. debugging) ensured the prototype system has been programmed correctly.

After the completion of this prototype development, the evaluation focus has been on success evaluation, which demonstrated successfulness of the actual support by answering the previously raised research questions and the hypotheses, thus experimentally indicated the usability and applicability once again. It was carried out according to an evaluation plan, which was defined based on a set of previously derived measurable success criteria and implemented with two use cases: cost estimation and FEA. The case of cost estimation mainly reflected the feasibility and effectiveness, which emphasised on the evaluation in regarding to knowledge acquisition and representation through the annotation anchors and contents, knowledge sharing, reasoning and reuse, and the knowledge modularity. The FEA case more focused on evaluating knowledge modelling methodology, the generality and the extendibility of such system.

9.7 Contribution Remarks

In conclusion, this framework represented in this thesis provides some fundamental contributions remarked as following, which have met the original aim and objectives.

- An in-depth understanding of the PLM, engineering design, CAD and MEV.
- A comprehensive review on the related technologies and the applications in various domains, with an emphasis on mechanical engineering. The outputs include a classification of annotation approaches and research gaps in ontological technologies.
- A novel systematic ontology-enabled annotation system in CAD with following important features:
  - Inherit and extend the ability of CAD system through a novel semantic mechanism to persistently associate design objects and a knowledge base with full granularities, including the coarse anchors such as bodies, and the fine ones such as faces and edges.
  - A coherent object-oriented multiple-viewpoint knowledge-based system for knowledge sharing and evolution;

  - A unified and standard-compliant formal metadata scheme for data interoperability and standard queries;
  - Reasoning services based on engineering semantics for automation;
  - Stand-off annotation strategy for maintenance and portability.
- A formal knowledge modelling methodology to suit different situations with regulated guidelines for further generality and extendibility, thus complete this approach as a sound and mature framework.
- A successful prototype to demonstrate framework and evaluate feasibility, effectiveness and generality with two cases: cost estimation and FEA.

In a nutshell, this is a first general purpose formal annotation framework based on CAD systems with full manipulation granulatiries, which is driven by a MEV knowledge base consisting of haierachical ontologies. This framework provides tight association between design object and engineering knowledge and opens possibility for downstream process and evolution.

To the research community, it has provided a foundation for engineering knowledge-based systems. Researchers can benefit from this formal object-oriented metadata scheme, explore on application of reasoning services and other aspects to further improve the process automation. To application vendors, this framework has provided a novel support for knowledge base. Thus, CAD vendors may consider upgrate from traditional knowledge bases to integral semantic knowledge bases in order to benefit from a closed loop of knowledge and information, and advanced reasoning services. On the other hand, other CAE application vendors may benefit from breaking the barrier of data format and tool integration, and thus to develop knowledge-based applications base on existing sophisiticated CAD systems in order to aid downstream engineering tasks.

References

3SL. (2009, 13 Jan 2009). "Cradle REQ." Retrieved 13 Jan, 2009, from http://www.threesl.com/pages/products/req.php.

Adams, V. (2006) "Do FEA Tools Give The Same Answers? A Comparison of Finite Element Analysis Software."

Adobe Systems Incorporated. (2009). "Features of Adobe Acrobat 9 Pro." Retrieved 09 January, 2009, from http://www.adobe.com/products/acrobatpro/features/.

Agbodan, D., D. Marcheix, G. Pierra and C. Thabaud (2003). A topological entity matching technique for geometric parametric models. Shape Modeling International, 2003.

Alatalo, P., J. Hyysalo, V. Mettovaara, P. Salonpää, P. Kuvaja, S. Heinonen, J. Kääriäinen, S. Soininen and M. Tihinen (2007). Collab Tools Report, The Merlin Consortium.

Alavi, M. and D. E. Leidner (2001). "Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues." MIS Quarterly **25**(1): 107-136.

Alink, W. (2005). XIRAF: An XML-IR Approach to Digital Forensics. Database Group, Faculty Electrical Engineering, Mathematics, and Computer Science. Enschede, The Netherlands, University of Twente. **Master**.

Amelunxen, C., F. Klar, A. Königs, T. Rötschke and A. Schürr (2008). Metamodel-based tool integration with moflon. Proceedings of the 30th international conference on Software engineering. Leipzig, Germany, ACM.

American National Standards Institute (1986). Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII). New York, American National Standards Institute.

Anand, V. B. (1993). Computer Graphics and Geometric Modeling for Engineers, John Wiley & Sons, Inc.

Andrea, M., R. Francesco, T. Maurizio and M. Salvatore (2008). Formalizing Knowledge by Ontologies: OWL and KIF. Pisa, Italy, The Institute of Informatics and Telematics of CNR.

ANSYS Inc. (2011). "ANSYS." from http://www.ansys.com/.

Anthony, L., W. C. Regli, J. E. John and S. V. Lombeyda (2001). "An Approach to Capturing Structure, Behavior, and Function of Artifacts in Computer-Aided Design." Journal of Computing and Information Science in Engineering **1**(2): 186-192.

Arpírez, J. C., O. Corcho, M. Fernández-López and A. Gómez-Pérez (2001). WebODE: a scalable workbench for ontological engineering. Proceedings of the 1st international conference on Knowledge capture. Victoria, British Columbia, Canada, ACM**:** 6-13.

Asbury, T. M., M. Mitman, J. Tang and W. J. Zheng (2010). "Genome3D: a viewer-model framework for integrating and visualizing multi-scale epigenomic information within a three-dimensional genome." Bmc Bioinformatics **11**: 444.

Asiedu, Y. and P. Gu (1998). "Product life cycle cost analysis: state of the art review." International Journal of Production Research **36**(4): 883 - 908.

ASME (2003). Y14.41 - 2003 Digital Product Definition Data Practices, ASME.

Aubry, S., I. Thouvenin, D. Lenne and J. Olive (2007). A knowledge model to read 3D annotations on a virtual mock-up for collaborative design. Proceedings of the 2007 11th International Conference on Computer Supported Cooperative Work in Design.

Ault, H. K. (1999). "3-D Geometric Modeling for the 21st Century." Engineering Design Graphics Journal **63**(2).

Autodesk Inc (2011a). AutoCAD 2012. DXF Reference, Autodesk Inc.

Autodesk Inc. (2011b). "Autodesk Inventor Products." Retrieved 06 June, 2011, from http://usa.autodesk.com/autodesk-inventor/.

Baba-Ali, M., D. Marcheix and X. Skapin (2009). "A Method To Improve Matching Process by Shape Characteristics in Parametric Systems." Computer-Aided Design and Applications **6**(3): 341-350.

Babic, B., N. Nesic and Z. Miljkovic (2008). "A review of automated feature recognition with rule-based pattern recognition." Computers in Industry **59**(4): 321-337.

Ball, A., L. Ding and M. Patel (2008). "An approach to accessing product data across system and software revisions." Advanced Engineering Informatics **22**(2): 222-235.

Batres, R., M. West, D. Leal, D. Price, K. Masaki, Y. Shimada, T. Fuchino and Y. Naka (2007). "An upper ontology based on ISO 15926." Computers & Chemical Engineering **31**(5-6): 519-534.

Bechhofer, S., R. Möller and P. Crowther (2003). The DIG Description Logic Interface. Description Logics.

Beckett, D. (2004, 10 February 2004). "RDF/XML Syntax Specification." Retrieved 23 May, 2008, from http://www.w3.org/TR/rdf-syntax-grammar/.

Beetz, J., v. J. Leeuwen and d. B. Vries (2005). An Ontology Web Language Notation of the Industry Foundation Classes. The 22nd CIB W78 Conference on Information Technology in Construction. Dresden, Germany, Technical University of Dresden**:** 193-198.

Beetz, J., J. van Leeuwen and B. de Vries (2009). "IfcOWL: A case of transforming EXPRESS schemas into ontologies." Artificial Intelligence for Engineering Design, Analysis and Manufacturing **23**(Special Issue 01): 89-101.

Bell, G., A. Parisi and M. Pesce. (1995). "The Virtual Reality Modeling Language Version 1.0 Specification." Retrieved 21 December, 2008, from http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html.

Bernaras, A., I. Laresgoiti and J. M. Corera (1996). Building and Reusing Ontologies for Electrical Network Applications. Proceedings of the 12th European Conference on Artificial Intelligence. Budapest, Hungary.

Berners-Lee, T. (2000). "Semantic Web - XML2000." Retrieved 09 July, 2011, from http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html.

Berners-Lee, T., R. T. Fielding and L. Masinter. (2005). "Uniform Resource Identifier (URI): Generic Syntax." Retrieved 21 June, 2011, from http://labs.apache.org/webarch/uri/rfc/rfc3986.html.

Bertram, S., M. Boniface, M. Surridge, N. Briscombe and M. Hall-May (2010). On-Demand Dynamic Security for Risk-Based Secure Collaboration in Clouds. Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on.

Bidarra, R., P. J. Nyirenda and W. F. Bronsvoort (2005). "A feature-based solution to the persistent naming problem." Computer-Aided Design & Applications **2**(1-4): 517-526.

Bilasco, I. M., J. Gensel, M. Villanova-Oliver and H. Martin (2005). On indexing of 3D scenes using MPEG-7. Proceedings of the 13th annual ACM international conference on Multimedia. Hilton, Singapore, ACM.

Bilasco, I. M., J. Gensel, M. Villanova-Oliver and H. Martin (2006). An MPEG-7 framework enhancing the reuse of 3D models. Proceedings of the eleventh international conference on 3D web technology. Columbia, Maryland, ACM.

Björk, B.-C. and M. Laakso (2010). "CAD standardisation in the construction industry -- A process view." Automation in Construction **19**(4): 398-406.

Blessing, L. and A. Chakrabarti (2009). DRM, a Design Research Methodology, Springer.

Blocher, E. C. m. (2005). Cost management : a strategic emphasis. Boston, Mass. ; London, McGraw-Hill/Irwin.

Bock, C., X. Zha, H.-w. Suh and J.-H. Lee (2010). "Ontological product modeling for collaborative design." Advanced Engineering Informatics **24**(4): 510-524.

Bond, A. H. and R. J. Ricci (1992). "Cooperation in aircraft design." Research in Engineering Design **4**(2): 115-130.

Bonino, D., F. Corno and L. Farinetti (2003). "DOSE: a distributed open semantic elaboration platform." Politecnico di Torino, Dip. di Automatica e Informatica.

Bracewell, R. H., K. Shea, P. M. Langdon, L. T. M. Blessing and P. J. Clarkson (2001). "A methodology for computational design tool research." Design Research - Theories, Methodologies, and Product Modelling: 181-188.

Bratt, S. (2007) "Semantic Web, and Other Technologies to Watch."

Bray, T. (1998). "RDF and Metadata." Retrieved 06 January, 2009, from http://www.xml.com/pub/a/98/06/rdf.html.

Bray, T., J. Paoli, C. M. Sperberg-McQueen, E. Maler and F. Yergeau. (2008, 26 November 2008). "Extensible Markup Language (XML) 1.0." Retrieved 21 December, 2010, from http://www.w3.org/TR/xml/.

Brickley, D. and R. V. Guha (2004). RDF Vocabulary Description Language 1.0: RDF Schema, World Wide Web Consortium.

Brinkkemper, S. (1996). "Method engineering: engineering of information systems development methods and tools." Information and Software Technology **38**(4): 275-280.

Brookes, N. J. and C. J. Backhouse (1998). "Understanding concurrent engineering implementation: a case-study approach." International Journal of Production Research **36**: 3035-3054.

Brousseau, E., S. Dimov and R. Setchi (2008). "Knowledge acquisition techniques for feature recognition in CAD models." Journal of Intelligent Manufacturing **19**(1): 21-32.

Brown, J. (2003) "Product Lifecycle Management Proving Value at Heinz: A PLM Case Study from the Consumer Goods Industry -- Food and Beverage."

Brown, K. N., J. H. Sims, Williams and C. A. McMahon (1992). Grammars of features in design, Pittsburgh, PA, USA, Publ by Kluwer Academic Publishers Group.

Brunnermeier, S. and S. A. Martin (1999). Interoperability Cost Analysis of the U.S. Automotive Supply Chain. Final Report. North Carolina, National Institute of Standards and Technology.

Brush, A. J. B., D. Bargeron, A. Gupta and J. J. Cadiz (2001). Robust annotation positioning in digital documents. Proceedings of the SIGCHI conference on Human factors in computing systems. Seattle, Washington, United States, ACM.

Buneman, P., R. Bose and D. Ecklund (2005). Annotation in Scientific Data: a Scoping Report (Draft). Edinburgh, University of Edinburgh**:** 1-12.

Burkett, M., J. Kemmeter and K. O'Marah (2002) "Product Lifecycle Management: What's Real Now." AMR Research Report.

CADAZZ. (2004). "Free CAD Software." Retrieved 06 June, 2011, from http://www.cadazz.com/free-cad-software.htm.

Canciglieri, O. J. and R. I. M. Young (2003). "Information sharing in multiviewpoint injection moulding design and manufacturing." International Journal of Production Research **41**(7): 1565-1586.

Cavalieri, S., P. Maccarrone and R. Pinto (2004). "Parametric vs. neural network models for the estimation of production costs: A case study in the automotive industry." International Journal of Production Economics **91**(2): 165-177.

Cera, C. D., W. C. Regli, I. Braude, Y. Shapirstein and C. V. Foster (2002). "A collaborative 3D environment for authoring design semantics." Computer Graphics and Applications, IEEE **22**(3): 43-55.

Chalupsky, H., R. M. MacGregor and T. Russ (2006). PowerLoom™ Manual. Marina Del Rey, University of Southern California.

Chaudhri, V. K., A. Farquhar, R. Fikes, P. D. Karp and J. P. Rice (1998). Open Knowledge Base Connectivity 2.0.3. Menlo Park, SRI International.

Cheung, W. M. and D. Schaefer (2010). Product Lifecycle Management: State-of-the-art and Future Perspectives. Enterprise Information Systems for Business Integration

in SMEs: Technological, Organizational and Social Dimensions. M. M. C. Cunha. Barcelos, Portugal, Poltechnic Institute of Cavado and Ave.
CIMdata (2002) "Product Lifecycle Management: Empowering the future of business."
Ciocoiu, M., D. S. Nau and M. Gruninger (2001). "Ontologies for Integrating Engineering Applications." Journal of Computing and Information Science in Engineering **1**(1): 12-22.
Clark & Parsia. (2011). "Pellet: OWL 2 Reasoner for Java." Retrieved 29 Jan, 2012, from http://clarkparsia.com/pellet.
Clough, R. W. (1960). The finite element method in plane stress analysis. ASCE 2nd Conference on Electronic Computation, Pittsburgh, PA.
Colombo, G., A. Mosca and F. Sartori (2007). "Towards the design of intelligent CAD systems: An ontological approach." Advanced Engineering Informatics **21**(2): 153-168.
COMSOL. (2011). "COMSOL Multiphysics®." from http://www.comsol.com/products/multiphysics/.
Connolly, D., F. v. Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider and L. A. Stein (2001). DAML+OIL (March 2001) Reference Description, World Wide Web Consortium.
Cook, R. D. (2007). Concepts and applications of finite element analysis.
Corcho, O., M. Fernández-López and A. Gómez-Pérez (2003). "Methodologies, tools and languages for building ontologies. Where is their meeting point?" Data & Knowledge Engineering **46**(1): 41-64.
Corcho, O., M. Fernández-López and A. Gómez-Pérez (2006). Ontological Engineering: Principles, Methods, Tools and Languages. Ontologies for Software Engineering and Software Technology. C. Calero, F. Ruiz and M. Piattini, Springer Berlin Heidelberg**:** 1-48.
Corcho, O. and A. Gomez-Perez (2000). "A roadmap to ontology specification languages." Knowledge Engineering and Knowledge Management, Proceedings **1937**: 80-96.
Cottrell, J. A., T. J. R. Hughes and Y. Bazilevs (2009). Isogeometric Analysis: Toward Integration of CAD and FEA. West Sussex, United Kingdom, John Wiley & Sons, Ltd.
Creswell, J. W. (2009). Research design : qualitative, quantitative, and mixed methods approaches. Los Angeles ; London, SAGE.
Cutkosky, M. R., R. S. Engelmore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, W. S. Mark, J. M. Tenenbaum and J. C. Weber (1993). "PACT: an experiment in integrating concurrent engineering systems." Computer **26**(1): 28-37.
Dartigues, C., P. Ghodous, M. Gruninger, D. Pallez and R. Sriram (2007). "CAD/CAPP integration using feature ontology." Concurrent Engineering-Research and Applications **15**(2): 237-249.
Dassault Systèmes. (2011a). "CATIA® - Virtual Design for Product Excellence." Retrieved 06 June, 2011, from http://www.3ds.com/products/catia/welcome/.
Dassault Systèmes. (2011b). "ENOVIA - Collaborative Product Lifecycle Management." Retrieved 01 June, 2011, from http://www.3ds.com/products/enovia/welcome/.
Dassault Systèmes. (2011c). "SIMULIA® for Realistic Simulation." Retrieved 06 June, 2011, from http://www.3ds.com/products/simulia/overview/.
Davies, D. (2008). Representation of multiple engineering viewpoints in Computer Aided Design through computer-interpretable descriptive markup. Mechanical Engineering. University of Bath, Bath, UK, University of Bath.
Davies, D. and C. A. McMahon (2006). "Multiple Viewpoint Design Modelling Through Semantic Markup." Proceedings of IDETC/CIE 2006.
DeRose, S. (2004). Markup Overlap: A Review and a Horse. Extreme Markup Languages 2004.
Devedžić, V. (2006). Introduction to the Semantic Web. Semantic Web and Education, Springer US. **12:** 29-69.

Ding, L., D. Davies and C. McMahon (2009). "The integration of lightweight representation and annotation for collaborative design representation." Research in Engineering Design **19**(4): 223-238.

Ding, L., P. Kolari, Z. Ding and S. Avancha (2007). Using Ontologies in the Semantic Web: A Survey. Ontologies. R. Sharman, R. Kishore and R. Ramesh, Springer US. **14:** 79-113.

Ding, L. and Y. Yue (2004). "Novel ANN-based feature recognition incorporating design by features." Comput. Ind. **55**(2): 197-222.

Drummond, N., S. Jupp, G. Moulton and R. Stevens (2009). A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.2, The University Of Manchester.

Duffy, A. H. B. and F. J. O'Donnell (1999). A design research approach. Critical Enthusiasm - Contributions to Design Science. Trondheim**:** 33 - 44.

Durkin, J. (1994). Expert systems: design and development, Macmillan.

Duverlie, P. and J. M. Castelain (1999). "Cost Estimation During Design Step: Parametric Method versus Case Based Reasoning Method." The International Journal of Advanced Manufacturing Technology **15**(12): 895-906.

Dzbor, M., E. Motta and J. Domingue (2004). "Opening Up Magpie via Semantic Services."

Eck, O. and D. Schaefer (2011). "A semantic file system for integrated product data management." Advanced Engineering Informatics **In Press, Corrected Proof**.

Egaña, M., E. Antezana and R. Stevens (2008). Transforming the Axiomisation of Ontologies: The Ontology Pre-Processor Language. OWLed DC.

Eiter, T., G. Ianni, A. Polleres, R. Schindlauer and H. Tompits (2006). Reasoning with Rules and Ontologies. Reasoning Web. P. Barahona, F. Bry, E. Franconi, N. Henze and U. Sattler, Springer Berlin / Heidelberg. **4126:** 93-127.

El-Mehalawi, M. and R. Allen Miller (2003a). "A database system of mechanical components based on geometric and topological similarity. Part I: representation." Computer-Aided Design **35**(1): 83-94.

El-Mehalawi, M. and R. Allen Miller (2003b). "A database system of mechanical components based on geometric and topological similarity. Part II: indexing, retrieval, matching, and similarity assessment." Computer-Aided Design **35**(1): 95-105.

Enrico, M. (1998). An Overview of the OCML Modelling Language, unknown.

Falconer, S. M., N. F. Noy and M.-A. Storey (2007). Ontology Mapping - A User Survey. Proc. ICSW workshop on Ontology Matching Busan, Korea.

Farish, M. (2008). Joined-up planning. Engineering and Technology, The Institution of Engineering and Technology**:** 61 - 63.

Farquhar, A., R. Fikes and J. Rice (1997). "The Ontolingua Server: a tool for collaborative ontology construction." International Journal of Human-Computer Studies **46**(6).

Fei, T. L. (2002). "Hypertext versus Knowledge Management: Data - Information - Knowledge." Retrieved 11 June, 2011, from http://www.cyberartsweb.org/cpace/ht/thonglipfei/data_info.html.

Fensel, D., M. Kerrigan and M. Zaremba (2008). Reasoning. Implementing Semantic Web Services. D. Fensel, M. Kerrigan and M. Zaremba, Springer Berlin Heidelberg**:** 137-165.

Fernandez-Lopez, M., A. Gomez-Perez and N. Juristo (1997). Methontology: From Ontological Art Towards Ontological Engineering. Proceedings of the AAAI97 Spring Symposium Stanford, USA.

Fikes, R., P. Hayes and I. Horrocks (2004). "OWL-QL - a language for deductive query answering on the Semantic Web." Web Semant. **2**(1): 19-29.

Finite Element Analysis Ltd. (2010). "LUSAS Analyst." from http://www.lusas.com/products/analyst.html.

Fokoue, A., A. Kershenbaum, L. Ma, E. Schonberg and K. Srinivas (2006). The Summary Abox: Cutting Ontologies Down to Size. The Semantic Web - ISWC 2006. I. Cruz, S. Decker, D. Allemanget al, Springer Berlin / Heidelberg. **4273:** 343-356.

Fuh, J. Y. H. and W. D. Li (2005). "Advances in collaborative CAD: the-state-of-the art." Computer-Aided Design **37**(5): 571-581.

FZI WIM and AIFB LS3. (2011). "KAON Tool Suit." Retrieved 22 June, 2011, from http://kaon.semanticweb.org/frontpage.

Gallaher, M. P., A. C. O'Connor, J. John L. Dettbarn and L. T. Gilday (2004). Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry. Maryland, NIST.

Galorath. (2008). "SEER for Manufacturing: Estimating Manufacturing Projects." Retrieved 24 July, 2009, from http://www.galorath.com/index.php/products/manufacturing/extended-capabilities-seer-mfg/.

Galorath Incorporated (2005). SEER-DFM Cost Designer for Parts, Process and Assembly: User's Manual. El Segundo, California, USA, Galorath Incorporated.

Gasevic, D., D. Djuric and V. Devedzic (2006). Model Driven Architecture and Ontology Development, Springer.

Gašević, D., D. Djurić and V. Devedžić (2006a). Knowledge Representation. Model Driven Architecture and Ontology Development**:** 3-43.

Gašević, D., D. Djurić and V. Devedžić (2006b). The Semantic Web. Model Driven Architecture and Ontology Development**:** 79-107.

Gero, J. S. (1990). "Design prototypes: a knowledge representation schema for design." AI Mag. **11**(4): 26-36.

Gero, J. S. and U. Kannengiesser (2004). "The situated function-behaviour-structure framework." Design Studies **25**(4): 373-391.

Giess, M. D., P. J. Wild and C. A. McMahon (2008). "The generation of faceted classification schemes for use in the organisation of engineering design documents." International Journal of Information Management **28**(5): 379-390.

Gomez-Perez, A., O. Corcho and M. Fernandez-Lopez (2004). Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition (Advanced Information and Knowledge Processing), Springer.

Grosse, I. R., J. M. Milton\, \#8211, benoit and J. C. Wileden (2005). "Ontologies for supporting engineering analysis models." Artif. Intell. Eng. Des. Anal. Manuf. **19**(1): 1-18.

Gruber, T. R. (1993). Towards Principles for the Design of Ontologies Used for Knowledge Sharing. Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers.

Gruber, T. R. (1995). "Toward principles for the design of ontologies used for knowledge sharing." International Journal of Human Computer Studies **43**(5-6): 907-907.

Grüninger, M. and C. Menzel (2003). "The process specification language (PSL) theory and applications." AI Mag. **24**(3): 63-74.

Guarino, N. (1998). "Formal ontology and information systems." Formal Ontology in Information Systems **46**: 3-15.

H'Mida, F., P. Martin and F. Vernadat (2006). "Cost estimation in mechanical production: The Cost Entity approach applied to integrated product engineering." International Journal of Production Economics **103**(1): 17-35.

Handschuh, S., S. Staab and F. Ciravegna (2002). "S-CREAM -- Semi-automatic CREAtion of Metadata."

Hendler, J. (2001). "Agents and the Semantic Web." IEEE Intelligent Systems **16**(2): 30-37.

Hevner, A., S. March, J. Park and S. Ram (2004). "Design Science in Information Systems Research." MIS Quarterly **28**(1).

Hewett, A. (2009). Product Lifecycle Management (PLM): Critical Issues and Challenges in Implementation
Information Technology and Product Development. S. Nambisan, Springer US. **5:** 81-105.
Hickson, I. (2011). "HTML5: A vocabulary and associated APIs for HTML and XHTML." Retrieved 21 June, 2011, from http://www.w3.org/TR/html5/.
Hisarciklilar, O. and J.-F. Boujut (2007). "Reducing the “Information Gap” Between Synchronous and Asynchronous Co-operative Design Phases."
Hoffmann, C. M. (2005). "Constraint-Based Computer-Aided Design." Journal of Computing and Information Science in Engineering **5**(3): 182-187.
Horrocks, I. (2005). Description Logic Reasoning. Manchester, UK, University of Manchester.
Horrocks, I., P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C.
Hung Ba, N., C. Bac, F. Silber-Chaussumier and L. Thang Quyet (2007). Towards Ontology-based Semantic File Systems. Research, Innovation and Vision for the Future, 2007 IEEE International Conference on.
Information Sciences Institute. (1998). "Loom Ontosaurus." Retrieved 22 June, 2011, from http://www.isi.edu/isd/ontosaurus.html.
ISO (1994a). ISO 10303-1:1994 Industrial automation systems and integration -- Product data representation and exchange. Part 1: Overview and fundamental principles.
ISO (1994b). ISO 10303-11:1994 Industrial automation systems and integration -- Product data representation and exchange. Part 11: Description methods: The EXPRESS language reference manual.
ISO (1994c). ISO 10303-21:1994 Industrial automation systems and integration -- Product data representation and exchange. Part 21: Implementation methods: Clear text encoding of the exchange structure.
ISO (1994d). ISO 10303-31:1994 Industrial automation systems and integration. Product data representation and exchange. Part 31: Conformance testing methodology and framework: general concepts.
ISO (1994e). ISO 10303-41:1994 Industrial automation systems and integration -- Product data representation and exchange. Part 41: Integrated generic resources: Fundamentals of product description and support.
ISO (1994f). ISO 10303-203:1994 Industrial automation systems and integration -- Product data representation and exchange. Part 203: Application protocol: Configuration controlled 3D designs of mechanical parts and assemblies.
ISO (2006). ISO 16792:2006 Technical product documentation -- Digital product definition data practices, BSI.
ISO (2008a). BS EN ISO 9001:2008 Quality management systems. Requirements, BSI.
ISO (2008b). ISO 24517-1:2008 Document management. Engineering document format using PDF. Use of PDF 1.6 (PDF/E-1), BSI.
ISO (2008c). ISO 32000-1:2008 Document management. Portable document format -- Part 1: PDF 1.7, BSI.
Iyer, G. R., J. J. Mills, S. Barber, V. Devarajan and S. Maitra (2006). "Using a Context-based Inference Approach to Capture Design Intent from Legacy CAD." Computer-Aided Design & Applications **3**(1-4): 269-278.
Jalote, P. (2005). An integrated approach to software engineering. New York, Springer.
Jasper, R. and M. Uschold. (1999). "A framework for understanding and classifying ontology applications." from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.83.7541.
Joo, J. and S. M. Lee (2009). "Adoption of the Semantic Web for overcoming technical limitations of knowledge management systems." Expert Systems with Applications **36**(3, Part 2): 7318-7327.
Jung, T., M. D. Gross and E. Y.-L. Do (2002). "Sketching annotations in a 3D Web environment."

Kahan, J. and M.-R. Koivunen (2001). Annotea: an open RDF infrastructure for shared Web annotations. Proceedings of the 10th international conference on World Wide Web Hong Kong.

Karp, P. D., V. K. Chaudhri and J. Thomere (1999). XOL: An XML-Based Ontology Exchange Language. Menlo Park, SRI International.

Kasik, D. J., W. Buxton and D. R. Ferguson (2005). "Ten CAD challenges." Computer Graphics and Applications, IEEE **25**(2): 81-92.

Khan, A. (2012, 24 Mar 2010). "Java Excel API - A Java API to read, write, and modify Excel spreadsheets." Retrieved 19 Jan, 2012, from http://jexcelapi.sourceforge.net/.

Kifer, M., G. Lausen and J. Wu (1995). "Logical-Foundations of Object-Oriented and Frame-Based Languages." Journal of the Association for Computing Machinery **42**(4): 741-843.

Kim, B., D. Mun and S. Han (2010). "Retrieval of CAD model data based on Web Services for collaborative product development in a distributed environment." The International Journal of Advanced Manufacturing Technology **50**(9): 1085-1099.

Kim, J., M. J. Pratt, R. G. Iyer and R. D. Sriram (2008). "Standardized data exchange of CAD models with design intent." Computer-Aided Design **40**(7): 760-777.

Kim, O., U. Jayaram, S. Jayaram and L. Zhu (2009). An Ontology Mapping Application Using a Shared Ontology Approach and a Bridge Ontology. Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2009, San Diego, California, USA, ASME.

Kiryakov, A., B. Popov, I. Terziev, D. Manov and D. Ognyanoff (2004). "Semantic annotation, indexing, and retrieval." Web Semantics: Science, Services and Agents on the World Wide Web **2**(1): 49-79.

Kitamura, Y., N. Washio, Y. Koji and R. Mizoguchi (2006). Towards Ontologies of Functionality and Semantic Annotation for Technical Knowledge Management. New Frontiers in Artificial Intelligence: Joint JSAI 2005 Workshop Post-Proceedings, Berlin / Heidelberg.

Korneffel, D. and P. Dvorak (2004). "Dimensions, tolerances, and more right on the model." Machine Design **76**(17): 70-72.

Krima, S., R. Barbau, X. Fiorentini, S. Rachuri, S. Foufou and R. D. Sriram (2009). OntoSTEP: OWL-DL ontology for STEP. International Conference on Product Lifecycle Management, Inderscience Enterprises Ltd.

Kusiak, A. and N. Larson (1999). Concurrent Engineering. Handbook of systems engineering and management. A. P. Sage, W. B. Rouse and Knovel (Firm). New York, Wiley**:** xix, 1236 p.

KwangHoon, L., C. A. McMahon and H. L. Kwan (2003). "Design of a feature-based multi-viewpoint design automation system." International Journal of CADCAM **3**: 67--75.

Lambrix, P., H. Tan, V. Jakoniene and L. Strömbäck (2007). Biological Ontologies. Semantic Web. C. J. O. Baker and K.-H. Cheung, Springer US**:** 85-99.

LAMP/IDE. (2008, 20 May 2008). "Standard for Exchange Product Data (STEP) " Retrieved 20 May, 2008, from http://www.tc184-sc4.org/SC4_Open/SC4%20Legacy%20Products%20(2001-08)/STEP_(10303)/.

Lee, J.-H. and H.-W. Suh (2007). "OWL-Based Product Ontology Architecture and Representation for Sharing Product Knowledge on a Web." ASME Conference Proceedings **2007**(48035): 853-861.

Lee, K. L., I. Kaymaz and C. A. McMahon (2001). The Use of Distributed Viewpoint-Dependent Feature-Based Modelling and the Response Surface Method in Design Assessment. Proceedings International Conference on Engineering Design, ICED01. Glasgow, Professional Engineering Publishing.

Li, C. (2012, 16 Jan 2012). "OntoCAD digital resources." from http://dl.dropbox.com/u/45907729/OntoCAD.rar.

Li, C., C. McMahon and L. Newnes (2009a). Annotation in Design Processes: Classification of Approaches. International Conference on Engineering Design, ICED'09, Stanford.

Li, C., C. McMahon and L. Newnes (2009b). Annotation in Product Lifecycle Management: A Review of Approaches. 29th Computers and Information in Engineering Conference (CIE), San Diego, ASME.

Li, Z., M. Kokkolaras, P. Papalambros and S. J. Hu (2008). "Product and Process Tolerance Allocation in Multistation Compliant Assembly Using Analytical Target Cascading." Journal of Mechanical Design **130**(9): 091701.

Li, Z., M. Liu, D. C. Anderson and K. Ramani (2005). Semantics-Based Design Knowledge Annotation and Retrieval. ASME 2005 International Design Engineering Technical Conferences & Computer and Information in Engineering Conference, Long Beach, California, USA.

Lim, S. C. J., Y. Liu and W. B. Lee (2009). Faceted search and retrieval based on semantically annotated product family ontology. Proceedings of the WSDM '09 Workshop on Exploiting Semantic Annotations in Information Retrieval, Barcelona, Spain, ACM.

Liu, Y.-J., D.-L. Zhang and M. M.-F. Yuen (2010). "A survey on CAD methods in 3D garment design." Computers in Industry **61**(6): 576-593.

Livermore Software Technology Corp. (2011). "LS-DYNA." from http://www.lstc.com/lsdyna.htm.

Living Paper. (2007). "Anoto functionality." Retrieved 10 January, 2009, from http://www.living-paper.com/anoto_functionality.html.

Long, J. A. (2000). Parametric cost estimating in the new millennium, PRICE Systems L.L.C.

Lortal, G., M. Lewkowicz and A. Todirascu-Courtier (2006). Annotations: A Way to Capture Experience. Knowledge-Based Intelligent Information and Engineering Systems. B. Gabrys, R. Howlett and L. Jain, Springer Berlin / Heidelberg. **4251:** 1131-1138.

Luke, S. and J. Heflin (2000). SHOE 1.01: Proposed Specification.

Maass, S. and J. Döllner (2006). Efficient View Management for Dynamic Annotation Placement in Virtual Landscapes. Smart Graphics. A. Butz, B. Fisher, A. Krüger and P. Olivier, Springer Berlin / Heidelberg. **4073:** 1-12.

MacGregor, R. M. (1991). "Inside the LOOM description classifier." SIGART Bull. **2**(3): 88-92.

Mahalingam, A., R. Kashyap and C. Mahajan (2010). "An evaluation of the applicability of 4D CAD on construction projects." Automation in Construction **19**(2): 148-159.

Manola, F. and E. Miller (2004). RDF Primer, World Wide Web Consortium.

Mao, M., Y. Peng and M. Spring (2010). "An adaptive ontology mapping approach with neural network based constraint satisfaction." Web Semantics: Science, Services and Agents on the World Wide Web **8**(1): 14-25.

Marcheix, D. and G. Pierra (2002). A survey of the persistent naming problem. Proceedings of the seventh ACM symposium on Solid modeling and applications. Saarbrucken, Germany, ACM**:** 13-22.

Marshall, C. C. (1997). "Annotation: from paper books to the digital library."

Martínez, J. M. (2004). "MPEG-7 Overview (version 10)." Retrieved 19 September 2008, 2008, from http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm.

Matsokis, A. and D. Kiritsis (2010). "An ontology-based approach for Product Lifecycle Management." Computers in Industry **61**(8): 787-797.

McGuinness, D. and Dieter (2003). Ontologies Come of Age. The Semantic Web: Why, What, and How, MIT Press.

McMahon, C. and J. Browne (1998). CADCAM : principles, practice, and manufacturing management. Harlow, Pearson Education.

McMahon, C., A. Lowe and S. Culley (2004). "Knowledge management in engineering design: personalization and codification." Journal of Engineering Design **15**(4): 307 - 325.

Melton, J. and A. R. Simon (1992). Understanding the New SQL: A Complete Guide. San Francisco, Morgan Kaufmann.

Microsoft Corporation. (2009). "Microsoft Office Word 2007 product overview." Retrieved 09 January, 2009, from http://office.microsoft.com/en-gb/word/HA101656411033.aspx.

Mindswap (2005). SWOOP - A Hypermedia-based Featherweight OWL Ontology Editor, Mindswap.

Ming, X. G., J. Q. Yan, W. F. Lu and D. Z. Ma (2005). "Technology solutions for collaborative product lifecycle management - Status review and future trend." Concurrent Engineering Research and Applications **13**(4): 311-319.

Moaveni, S. (1999). Finite element analysis : theory and application with ANSYS. Upper Saddle River, N.J., Prentice Hall.

Mont, O. (2001). Introducing and developing a Product-Service System (PSS) concept in Sweden.

Motik, B. (2011). "KAON2." Retrieved 22 June, 2011, from http://kaon2.semanticweb.org/.

Mun, D. and S. Han (2005). "Identification of Topological Entities and Naming Mapping for Parametric CAD Model Exchanges." International Journal of CAD/CAM **5**(1).

NEi Software. (2011). "NEi Nastran: Advanced finite element analysis." from http://www.nenastran.com/nei-nastran.php.

Newnes, L. B., A. R. Mileham and H. Hosseini-Nasab (2007). "On-screen real-time cost estimating." International Journal of Production Research **45**(7): 1577-1594.

Niazi, A., J. S. Dai, S. Balabani and L. Seneviratne (2006). "Product Cost Estimation: Technique Classification and Methodology Review." Journal of Manufacturing Science and Engineering **128**(2): 563-575.

NIST. (2011). "The Initial Graphics Exchange Specification (IGES)." Retrieved 08 June, 2011, from http://ts.nist.gov/standards/iges/.

O'Connor, M. and A. Das (2009). SQWRL: a Query Language for OWL. Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2009), Chantilly, VA, United States, CEUR-WS.org.

ontoprise GmbH (2009). OntoStudio. Karlsruhe.

Oracle. (2011). "Agile Product Lifecycle Management Solutions." Retrieved 01 June, 2011, from http://www.oracle.com/us/products/applications/agile/index.html.

OuYang, C. and T. S. Lin (1997). "Developing an integrated framework for feature-based early manufacturing cost estimation." International Journal of Advanced Manufacturing Technology **13**(9): 618-629.

Ovsiannikov, I. A., M. A. Arbib and T. H. McNeill (1999). Annotation technology. Int. J. Human-Computer Studies.

Ozbayrak, M., M. Akgun and A. K. Turker (2004). "Activity-based cost estimation in a push/pull advanced manufacturing system." International Journal of Production Economics **87**(1): 49-65.

Pahl, G., W. Beitz, J. Feldhusen and K. H. Grote (2007). Engineering Design: A Systematic Approach, Springer.

Pahl, G., W. Beitz and K. Wallace (1984). Engineering design. London, Design Council.

Parametric Technology Corporation. (2011a). "Creo Elements/Pro." from http://www.ptc.com/products/creo-elements-pro/.

Parametric Technology Corporation. (2011b). "Windchill." Retrieved 01 June, 2011, from http://www.ptc.com/products/windchill/.

Patil, L., D. Dutta and R. Sriram (2005). "Ontology-based exchange of product data semantics." Automation Science and Engineering, IEEE Transactions on **2**(3): 213-225.

PDES Inc. (1998). Recommended Practices for AP 203.

Peachavanish, R., H. A. Karimi, B. Akinci and F. Boukamp (2006). "An ontological engineering approach for integrating CAD and GIS in support of infrastructure management." Adv. Eng. Inform. **20**(1): 71-88.

Peak, R. S., J. Lubell, V. Srinivasan and S. C. Waterbury (2004). "STEP, XML, and UML: Complementary technologies." Journal of Computing and Information Science in Engineering **4**(4): 379-390.

Pirró, G. and D. Talia (2010). "UFOme: An ontology mapping system with strategy prediction capabilities." Data & Knowledge Engineering **69**(5): 444-471.

Pittarello, F. and A. D. Faveri (2006). "Semantic Description of 3D Environments : a Proposal Based on Web Standards."

PLM Technology Guide. (2008). "PLM Systems." Retrieved 31 May, 2009, from http://plmtechnologyguide.com/site/?page_id=27.

Polanyi, M. (1966). The Tacit Dimension. London, Routledge & Kegan Paul.

Posada, J., C. Toro, S. Wundrak and A. Stork (2005). Ontology Modelling of Industry Standards for Large Model Visualization and Design Review using Protégé. 8th Intl. Protégé Conference. Madrid, Spain.

Posada, J., C. Toro, S. Wundrak and A. Stork (2006). "Using ontologies and STEP standards for the semantic simplification of CAD models in different engineering domains." Applied Ontology **1**(3): 263-279.

Pratt, M. J. (2001). "Introduction to ISO 10303---the STEP Standard for Product Data Exchange." Journal of Computing and Information Science in Engineering **1**(1): 102-103.

Pratt, M. J. (2005). "ISO 10303, the STEP standard for product data exchange, and its PLM capabilities." International Journal of Product Lifecycle Management **1**: 86-94.

Prud'hommeaux, E. and A. Seaborne (2008). SPARQL Query Language for RDF.

Qian, L. and J. S. Gero (1996). "Function-behavior-structure paths and their role in analogy-based design." Ai Edam-Artificial Intelligence for Engineering Design Analysis and Manufacturing **10**(4): 289-312.

Qing, Y. (1995). "The Standardization of Mechanical Engineering in Qin Dynasty." Journal of Northwest Sci-Tech University of Agriculture and Forestry **23**(S1).

Rachuri, S., E. Subrahmanian, A. Bouras, S. J. Fenves, S. Foufou and R. D. Sriram (2008). "Information sharing and exchange in the context of product lifecycle management: Role of standards." Comput. Aided Des. **40**(7): 789-800.

Rameshbabu, V. and M. S. Shunmugam (2009). "Hybrid feature recognition method for setup planning from STEP AP-203." Robotics and Computer-Integrated Manufacturing **25**(2): 393-408.

Rangan, R. M., S. M. Rohde, R. Peak, B. Chadha and P. Bliznakov (2005). "Streamlining product lifecycle processes: A survey of product lifecycle management implementations, directions, and challenges." Journal of Computing and Information Science in Engineering **5**(3): 227-237.

Reeves, G. A., D. Talavera and J. M. Thornton (2009). "Genome and proteome annotation: organization, interpretation and integration." Journal of the Royal Society, Interface / the Royal Society **6**(31): 129-47.

Robertson, B. F. and D. F. Radcliffe (2009). "Impact of CAD tools on creative problem solving in engineering design." Computer-Aided Design **41**(3): 136-146.

Rolland, C. (1998). A comprehensive view of process engineering. Advanced Information Systems Engineering**:** 1-24.

Roy, R., S. Kelvesjo, S. Forsberg and C. Rush (2001). "Quantitative and qualitative cost estimating for engineering design." Journal of Engineering Design **12**(2): 147-162.

Royce, W. (1970). Managing the Development of Large Software Systems. Proc. IEEE Wescon.

Saaksvuori, A. and A. Immonen (2008). Product Lifecycle Management, Springer.

Salzman, H. (1989). "Computer-Aided-Design - Limitations in Automating Design and Drafting." Ieee Transactions on Engineering Management **36**(4): 252-261.

Schlenoff, C., M. Gruninger, F. Tissot, J. Valois, J. Lubell and J. Lee (2000). The Process Specification Language (PSL) Overview and Version 1.0 Specification. Gaithersburg, National Institute of Standards and Technology, .

Schroeter, R., J. Hunter, J. Guerin, I. Khan and M. Henderson (2006). A synchronous multimedia annotation system for secure collaboratories, Piscataway, NJ 08855-1331, United States, Institute of Electrical and Electronics Engineers Computer Society.

SCRA (2006). STEP Application Handbook ISO 10303 Version 3. North Charleston, SCRA.

Setchi, R., Q. Tang and I. Stankov (2011). "Semantic-based information retrieval in support of concept design." Advanced Engineering Informatics **25**(2): 131-146.

Setchi, R. M. and Q. Tang (2007). "Concept Indexing using Ontology and Supervised Machine Learning." Proceedings of World Academy of Science, Engineering and Technology, Vol 19 **19**: 221-226.

Shaban-Nejad, A., C. Baker, V. Haarslev and G. Butler (2005). The FungalWeb Ontology: Semantic Web Challenges in Bioinformatics and Genomics. The Semantic Web – ISWC 2005. Y. Gil, E. Motta, V. Benjamins and M. Musen, Springer Berlin / Heidelberg. **3729:** 1063-1066.

Shah, J. J. and M. Mäntylä (1995). Parametric and feature-based CAD/CAM : concepts, techniques, and applications. New York, John Wiley & Sons, Inc.

Sharma, A. (2007). "The shift in sales organizations in business-to-business services markets." Journal of Services Marketing **21**(5): 326 - 333.

Shen, W., Q. Hao, H. J. Yoon and D. H. Norrie (2006). "Applications of agent-based systems in intelligent manufacturing: An updated review." Advanced Engineering Informatics **20**(4): 415-431.

Shen, Z., R. Issa and L. Gu (2007). Semantic 3D CAD and Its Applications in Construction Industry – An Outlook of Construction Data Visualization. Advances in Visual Information Systems. G. Qiu, C. Leung, X. Xue and R. Laurini, Springer Berlin / Heidelberg. **4781:** 461-467.

Shi, L. and R. Setchi (2010). An Ontology Based Approach to Measuring the Semantic Similarity between Information Objects in Personal Information Collections. Knowledge-Based and Intelligent Information and Engineering Systems. R. Setchi, I. Jordanov, R. Howlett and L. Jain, Springer Berlin / Heidelberg. **6276:** 617-626.

Shin, I., S. Kim, J. Busby, R. E. Hibberd and C. A. McMahon (2006). An Application of Semantic Annotations to Design Errors. 2006 International Conference on Hybrid Information Technology (ICHIT'06).

Siemens PLM Software Inc (2008). NX 6 Help Library.

Siemens PLM Software Inc. (2011a). "Explore NX and discover your solution." Retrieved 06 June, 2011, from http://www.plm.automation.siemens.com/en_us/products/nx/index.shtml.

Siemens PLM Software Inc. (2011b). "Explore Teamcenter and discover your solution." Retrieved 01 June, 2011, from http://www.plm.automation.siemens.com/en_us/products/teamcenter/.

Silberschatz, A., H. Korth and S. Sudarshan (2010). Database system concepts, McGraw-Hill.

Silcher, S., J. Minguez, T. Scheibler and B. Mitschang (2010). A service-based approach for next-generation Product Lifecycle Management. Information Reuse and Integration (IRI), 2010 IEEE International Conference on.

Skulmoski, G. J., F. T. Hartman and J. Krahn (2007). "The Delphi method for graduate research." Journal of Information Technology Education **Volume 6**.

Smith, M. K., C. Welty and D. L. McGuinness (2004). OWL Web Ontology Language Guide, World Wide Web Consortium.

Soanes, C. and A. Stevenson (2005a). annotation noun. Oxford Dictionary of English C. Soanes and A. Stevenson. Oxford, Oxford University Press.

Soanes, C. and A. Stevenson (2005b). methodology noun. Oxford Dictionary of English C. Soanes and A. Stevenson. Oxford, Oxford University Press.
SofTech Inc. (2011). "Product Lifecycle Management (PLM) Software Solutions: ProductCenter® PLM automates your product data and lifecycle processes." Retrieved 01 June, 2011, from http://www.softech.com/products/plm/.
Sommerville, I. and P. Sawyer (1997). "Viewpoints: principles, problems and a practical approach to requirements engineering." Annals of Software Engineering **3**(1): 101-130.
Song, H., F. Guimbretière, C. Hu and H. Lipson (2006). "ModelCraft: Capturing Freehand Annotations and Edits on Physical 3D Models."
Song, I. and S. Han (2010). Parametric CAD Data Exchange Using Geometry-Based Neutral Macro File
Cooperative Design, Visualization, and Engineering. Y. Luo, Springer Berlin / Heidelberg. **6240:** 145-152.
Sonnenwald, D. H. (1996). "Communication roles that support collaboration during the design process." Design Studies **17**(3): 277-301.
Soo, V.-W., C.-Y. Lee, C.-C. Li, S. L. Chen and C.-c. Chen (2003). Automated Semantic Annotation and Retrieval Based on Sharable Ontology and Case-based Learning Techniques. 2003 Joint Conference on Digital Libraries (JCDL' 2003).
Sowa, J. (1999). Knowledge Representation: Logical, Philosophical, and Computational Foundations, Course Technology.
Srinivasan, V. (2008). "Standardizing the specification, verification, and exchange of product geometry: Research, status and trends." Computer-Aided Design **40**(7): 738-749.
Staab, S., R. Studer, H.-P. Schnurr and Y. Sure (2001). "Knowledge Processes and Ontologies." IEEE Intelligent Systems **16**(1): 26-34.
Stanford Center for Biomedical Informatics Research (2011). Protégé. 4.0 beta. Stanford, Stanford University
Stanford Medical Informatics. (2010, 29 Oct 2010). "SQWRL." Retrieved 03 Feb, 2011, from http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL.
Stanford Medical Informatics. (2011, 13 Jan 2011). "SWRL Language FAQ." Retrieved 03 Feb, 2011, from http://protege.cim3.net/cgi-bin/wiki.pl?WikiHomePage.
Stanford University (2005). Ontolingua, Stanford University
Stark, J. (2011). Product Lifecycle Management, Springer London**:** 1-16.
STEP Tools Inc. (2012). "ST-Developer Tools Reference." Retrieved 17 Feb, 2012, from http://www.steptools.com/support/stdev_docs/devtools/index.html.
Stephan, G. s., H. s. Pascal and A. s. Andreas (2007). Knowledge Representation and Ontologies. Semantic Web Services. R. Studer, S. Grimm and A. Abecker, Springer Berlin Heidelberg**:** 51-105.
Strub, J. R., E. D. Rekow and S. Witkowski (2006). "Computer-aided design and fabrication of dental restorations - Current systems and future possibilities." Journal of the American Dental Association **137**(9): 1289-1296.
Sunil, V. B. and S. S. Pande (2008). "Automatic recognition of features from freeform surface CAD models." Computer-Aided Design **40**(4): 502-517.
Sutherland, I. (1964). Sketch pad a man-machine graphical communication system. DAC '64: Proceedings of the SHARE design automation workshop, ACM.
Swartout, B., R. Patil, K. Knight and T. Russ (1997). "Toward Distributed Use of Large-Scale Ontologies." AAAI Symposium on Ontological Engineering.
Swift, K. G. and J. D. Booker (2003). Process Selection: from design to manufacture, Elsevier Butterworth-Heinemann.
Tan, S. (2006) "User survey: Mechanical CAx. Europe and North America, 2005."
Tay, F. E. H. and A. Roy (2003). "CyberCAD: a collaborative approach in 3D-CAD technology in a multimedia-supported environment." Comput. Ind. **52**(2): 127-145.

Terzi, S., A. Bouras, D. Dutta, M. Garetti and D. Kiritsis (2010). "Product lifecycle management - from its history to its new role." International Journal of Product Lifecycle Management **4**(2010): 360 - 389.

Theodoulis, B., H. Karanikas, B. Black, J. McNaught, H. B. Slot, C. v. d. Touw, P. Nierop, R. v. d. Maas, D. Out, J. Ellman, G. P. Zarri, L. Bernard, G. Orphanos, C. Tsalidis, M. Spiliopoulou, M. Brunzel, F. Rinaldi, M. Hess, J. Dowdall, M. King, V. Sauron, N. Underwood, A. Lisowska, L. v. d. Plas, S. Taraviras, T. Mavroudakis and A. Persidis (2003). "Common annotation scheme." Information society technologies: 1 - 42

Thouvenin, I., A. Guenand, D. Lenne and S. Aubry (2005). Knowledge integration in early design stages for collaboration on a virtual mock up, Piscataway, NJ 08855-1331, United States, Institute of Electrical and Electronics Engineers Computer Society.

Tornincasa, S. and F. D. Monaco (2010). The Future and the Evolution of CAD. 14th International Research/Expert Conference "Trends in the Development of Machinery and Associated Technology", Mediterranean Cruise.

Toro, C., J. Posada, S. Wundrak and A. Stork (2006). "Improving Virtual Reality Applications in CAD through Semantics." The International Journal of Virtual Reality **5**(4): 39-46.

Umeda, Y., H. Takeda, T. Tomiyama and H. Yoshikawa (1990). Function, Behaviour, and Structure, Boston, Computational Mechanics Publications.

University of Manchester, Clark & Parsia LLC and University of Ulm. (2011). "The OWL API." Retrieved 19 Jan, 2012, from http://owlapi.sourceforge.net/index.html.

Uren, V., P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta and F. Ciravegna (2006). "Semantic annotation for knowledge management: Requirements and a survey of the state of the art." Web Semantics: Science, Services and Agents on the World Wide Web **4**(1): 14-28.

Uschold, M. and M. Gruninger (1996). "Ontologies: principles, methods and applications." The Knowledge Engineering Review **11**(02): 93-136.

Uschold, M. and M. King (1995). Towards a Methodology for Building Ontologies. International Joint Conference on Artificial Inteligence (IJCAI95), Workshop on Basic Ontological Issues in Knowledge Sharing.

Vargas-Vera, M., E. Motta, J. Domingue, M. Lanzoni, A. Stutt and F. Ciravegna (2002). "MnM : Ontology Driven Semi-automatic and Automatic Support for Semantic Markup."

Veeramani, D. and P. Joshi (1997). "Methodologies for rapid and effective response to requests for quotation (RFQs)." IIE Transactions **29**(10): 825-838.

Verhagen, W. J. C., P. Bermell-Garcia, R. E. C. van Dijk and R. Curran (2011). "A critical review of Knowledge-Based Engineering: An identification of research challenges." Advanced Engineering Informatics **In Press, Corrected Proof**.

Vijay, S. (2011). "An integration framework for product lifecycle management." Computer-Aided Design **43**(5): 464-478.

Vince, J. A. (2004). Introduction to virtual reality, Springer.

W3C. (2009). "World Wide Web Consortium (W3C)." Retrieved 21 December, 2010, from http://www.w3.org/Consortium/.

W3C Schools. (2008). "Introduction to XML." Retrieved 21 December, 2008, from http://www.w3schools.com/xml/xml_whatis.asp.

Wang, S. (2005). Annotation persistence over dynamic documents. Department of Civil and Enviromental Engineering, MIT. **Doctor of Philosophy**.

Wang, Y. and B. O. Nnaji (2006). "Document-Driven Design for Distributed CAD Services in Service-Oriented Architecture." Journal of Computing and Information Science in Engineering **6**: 127 - 138.

Web3D Consortium. (2008). "What is X3D?" Retrieved 21 December, 2008, from http://www.web3d.org/about/overview/.

Webster, D. E. (1988). "Mapping the design information representation terrain." Computer **21**(12): 8-23.

Weisemoller, I., M. Wieber and D. Stuckert (2008). MOFLON by example. A guided tour through your first project, moflon.org

Wiig, K. M. (1997). "Knowledge Management: An Introduction and Perspective." Journal of Knowledge Management **1**(1): 6 - 14.

Wikipedia®. (2011a). "Category:Free computer-aided design software." Retrieved 06 November, 2011, from http://en.wikipedia.org/wiki/Category:Free_computer-aided_design_software.

Wikipedia®. (2011b). "List of computer-aided design editors for architecture, engineering and construction." Retrieved 06 June, 2011, from http://en.wikipedia.org/wiki/List_of_computer-aided_design_editors_for_architecture,_engineering_and_construction.

Wild, P., M. Giess and C. McMahon (2009). "Describing engineering documents with faceted approaches: Observations and reflections." Journal of Documentation **65**(3): 420-445.

Woodwark, J. (1986). Computing shape : an introduction to the representation of component and assembly geometry for computer-aided engineering. London, Butterworths.

Yee, K.-P. (2002). CritLink: Advanced Hyperlinks Enable Public Annotation on the Web. CSCW 2002 conference, New Orleans.

Yoshioka, M., Y. Umeda, H. Takeda, Y. Shimomura, Y. Nomaguchi and T. Tomiyama (2004). "Physical concept ontology for the knowledge intensive engineering framework." Advanced Engineering Informatics **18**(2): 95-113.

Yu, J., J. Cha, Y. Lu, W. Xu and M. Sobolewski (2010). "A CAE-integrated distributed collaborative design system for finite element analysis of complex product based on SOOA." Advances in Engineering Software **41**(4): 590-603.

Zhan, P. (2007). An Ontology-Based Approach for Semantic Level Information Exchange and Integration in Applications for Product Lifecycle Management. School of Mechanical and Materials Engineering, Washington State University. **Doctor of Philosophy:** 135.

Zhou, Z. L., Z. Lu and J. Z. Gu (2008). "Towards an Ontology-based Content Security Scheme." Fifth International Conference on Fuzzy Systems and Knowledge Discovery, Vol 4, Proceedings: 385-390.

Zhu, L., U. Jayaram, S. Jayaram and O. Kim (2009). Ontology-Driven Integration of CAD/CAE Applications: Strategies and Comparisons. Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2009, San Diego, California, USA, ASME.

Zhu, L., U. Jayaram, S. Jayaram and O. Kim (2010). Querying and Reasoning With Product Engineering Ontologies: Moving Past Modeling. Proceedings of the ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2010 Montreal, Quebec, Canada, ASME.

Appendix 1: Full List of STEP Class 6 Entities

action
action_assignment
action_directive
action_method
action_request_assignment
action_request_solution
action_request_status
action_status
Address
advanced_brep_shape_representation
advanced_face
alternate_product_relationship
application_context
application_context_element
application_protocol_definition
approval
approval_assignment
approval_date_time
approval_person_organization
approval_relationship
approval_role
approval_status
area_measure_with_unit
area_unit
assembly_component_usage
assembly_component_usage_substitute
axis1_placement
axis2_placement_2d
axis2_placement_3d
b_spline_curve
b_spline_curve_with_knots
b_spline_surface
b_spline_surface_with_knots
bezier_curve
bezier_surface
boundary_curve
bounded_curve
bounded_pcurve
bounded_surface
bounded_surface_curve
brep_with_voids
calendar_date
cartesian_point
cartesian_transformation_operator
cartesian_transformation_operator_3d

cc_design_approval
cc_design_certification
cc_design_contract
cc_design_date_and_time_assignment
cc_design_person_and_organization_assignment
cc_design_security_classification
cc_design_specification_reference
certification
certification_assignment
certification_type
change
change_request
circle
closed_shell
composite_curve composite_curve_on_surface composite_curve_segment
configuration_design
configuration_effectivity
configuration_item
conic
conical_surface connected_edge_set connected_face_set context_dependent_shape_representation
context_dependent_unit
contract
contract_assignment
contract_type
conversion_based_unit
coordinated_universal_time_offset
curve
curve_bounded_surface curve_replica cylindrical_surface
date
date_and_time
date_and_time_assignment
date_time_role
dated_effectivity
definitional_representation degenerate_pcurve degenerate_toroidal_surface
design_context
design_make_from_relationship

dimensional_exponents
directed_action
direction
document
document_reference
document_relationship
document_type
document_usage_constraint
document_with_class
edge
edge_based_wireframe_model edge_based_wireframe_shape_representation
edge_curve
edge_loop
effectivity
elementary_surface
ellipse
evaluated_degenerate_pcurve
executed_action
face
face_bound
face_outer_bound
face_surface
faceted_brep faceted_brep_shape_representation founded_item functionally_defined_transformation geometric_curve_set
geometric_representation_context
geometric_representation_item
geometric_set
geometrically_bounded_surface_shape_representation
geometrically_bounded_wireframe_shape_representation
global_uncertainty_assigned_context
global_unit_assigned_context
hyperbola
intersection_curve
item_defined_transformation
length_measure_with_unit
length_unit
line
local_time
loop
lot_effectivity
manifold_solid_brep
manifold_surface_shape_representation
mapped_item

mass_measure_with_unit
mass_unit
measure_with_unit
mechanical_context
named_unit
next_assembly_usage_occurrence
offset_curve_3d
offset_surface
open_shell
ordinal_date
organization
organization_relationship
organizational_address
organizational_project
oriented_closed_shell
oriented_edge
oriented_face
oriented_open_shell
oriented_path
outer_boundary_curve
parabola
parametric_representation_context
path
pcurve
person
person_and_organization
person_and_organization_assignment
person_and_organization_role
personal_address
placement
plane
plane_angle_measure_with_unit
plane_angle_unit
point
point_on_curve point_on_surface point_replica poly_loop
polyline
product
product_category
product_category_relationship
product_concept
product_concept_context
product_context
product_definition
product_definition_context
product_definition_effectivity
product_definition_formation
product_definition_formation_with_specified_source
product_definition_relationship

product_definition_shape
product_definition_usage
product_definition_with_associated_documents
product_related_product_category
promissory_usage_occurrence
property_definition
property_definition_representation
quantified_assembly_component_usage
quasi_uniform_curve
quasi_uniform_surface
rational_b_spline_curve
rational_b_spline_surface
rectangular_composite_surface rectangular_trimmed_surface reparametrised_composite_curve_segment
representation
representation_context
representation_item
representation_map
representation_relationship representation_relationship_with_transformation seam_curve
security_classification
security_classification_assignment
security_classification_level
serial_numbered_effectivity
shape_aspect shape_aspect_relationship
shape_definition_representation
shape_representation
shape_representation_relationship shell_based_surface_model shell_based_wireframe_model shell_based_wireframe_shape_representation
si_unit
solid_angle_measure_with_unit
solid_angle_unit
solid_model
specified_higher_usage_occurrence
spherical_surface
start_request
start_work
supplied_part_relationship
surface
surface_curve
surface_of_linear_extrusion
surface_of_revolution
surface_patch surface_replica
swept_surface
topological_representation_item
toroidal_surface trimmed_curve
uncertainty_measure_with_unit
uniform_curve
uniform_surface
vector
versioned_action_request
vertex
vertex_loop
vertex_point
vertex_shell
volume_measure_with_unit
volume_unit
week_of_year_and_day_date
wire_shell

Note:
1. The table refers to p279 – p283 in ISO 10303-203:1994.
2. All entries are included in conformance Class 6 but not the shaded entries, which are included in other classes.

Appendix 2: Functionality Acceptance Test Specification

OntoCAD Functionality Acceptance Test Specification

University of Bath
Chunlei Li
Supervisors: Chris McMahon; Linda Newnes

Document Control

Title: OntoCAD Functionality Acceptance Test Specification

Document No: BATH/001/A/27.10.2011

Project Number:	RC-ME0096M	**Status:**	Draft/Definitive
Product Number:	OntoCAD	**File:**	OntoCAD Functionality Acceptance Test Specification.doc

Synopsis:
This document defines the functionality acceptance test for OntoCAD prototype system.

	Printed Name	**Signature**
Originator:	Chunlei Li	
	Printed Name	**Signature**
Authorisation:	Chris McMahon	
	Printed Name	**Signature**
Doc. Review:	Chris McMahon	
	Printed Name	**Signature**
QA. Review:	Linda Newnes	

Circulation List: Glen Mullineux
William Megill

All other copies are Uncontrolled

Change Control: Document must be reviewed by someone other than the Originator/Editor.

Retention Period: This issue should be held on file until MM/YYYY.

Changes History:

Issue:	**Date:**	**Changes:**
Bath/001/Ad/24.07.2011	24/07/2011	Original Issue

Contents

1 Introduction

This document is designed to carry out Functionality Acceptance Tests (FAT) for a prototype software application called OntoCAD prototype system, which is referred as OntoCAD from this point. OntoCAD aims to demonstrate and evaluate the concept proposed in the doctoral degree thesis. For the details of the concept, please refer to the thesis itself. This document will be mainly focused on the operation procedures for evaluation purposes.

1.1 System Requirement

Operating System	Microsoft Windows XP SP2 (SP3) or Windows 7 (32-bit)
Memory	1 GB
Language	English
Disk Space	At least 4.5 GB

1.2 Prerequisites

- Siemens PLM Software NX6;
- Java Runtime Environment version 1.6 or higher;
- Protégé version 4.1 Beta or higher;
- SEER-DFM version 6.0.15.

1.3 Installation

The objective in this section is to properly set up NX6 with OntoCAD prototype system including the ontologies.

1.3.1 Install menu files

1.3.1.1 Locate following Menuscript files in NX add-on directory, e.g. “E:\NX\addon\startup”:

- ONTOCAD.men
- ONTOCADWatchdog.men

1.3.1.2 Locate the following secondary level Menuscript file in NX add-on directory, e.g. “E:\NX\addon\application”:

- ONTOCADANNOTATIONS_APP.men

1.3.2 Install Java application files

1.3.2.1 Locate following files in NX add-on directory, e.g. “E:\NX\addon\application”:

- OntoCAD.jar
- OntoCADAddAnnotation.jar
- OntoCADWatchdog.jar

1.3.3 Install NX Open User Interface Styler files

1.3.3.1 Locate following files in NX add-on directory, e.g. “E:\NX\addon\application”:

- OntoCADAddAnnotation.dlg
- OntoCADCreateAnnotationContentDlg.dlg
- OntoCADFillDataPropertiesDlg.dlg
- OntoCADWDDlg.dlg

1.3.4 Install OWL ontology files

1.3.4.1 Locate following file in an ontology working directory, e.g. "C:\MechanicalEngineering":

- MechanicalEngineeringOntology.owl

2 Automatic Labelling

2.1 Objectives

The objectives in this section include verifying whether the OntoCAD can be loaded, the user interface appears as designed and automatically apply annotation anchor identifiers (AAI) to all bodies, faces and edges of a NX6 design part.

2.2 Operations

2.2.1 Load OntoCAD

2.2.1.1 Run NX6, click on "File" from toolbar menu bar, and then open towbar project.

2.2.1.2 Click on "Start" from toolbar standard, and then "All Applications". Check that "OntoCAD" is in the drop-down list and click on it.

2.2.1.3 Check that OntoCAD appears on toolbar menu bar.

2.2.2 Apply labels

2.2.2.1 Click on "OntoCAD" from toolbar menu bar, check that drop-down menu pops up, in which two items are contained: "Auto label" and "Add annotation".

2.2.2.2 Click on "Auto label" and check that an "OntoCAD Info" dialog pops up and indicates automatic labels are applied.

2.2.2.3 In the type filter drop-down menu select "Solid Body" and select the towbar body. Check that "Solid Body 'BODY_1' selected" appears in the information bar.

2.2.2.4 In the type filter drop-down menu select "Face" and select any face of the towbar. Check that "Face 'FACE_x' selected" appears in the information bar, where 'x' indicates the ID number.

2.2.2.5 In the type filter drop-down menu select "Edge" and select any edge of the towbar. Check that "Edge 'EDGE_x' selected" appears in the information bar, where 'x' indicates the ID number.

2.2.2.6 Save the modified NX part by clicking on "File → Save".

2.2.3 Check ontologies

2.2.3.1 Run Protégé ontology editor and click on "Open OWL ontology" option to load the

project “MechanicalEngineeringOntology.owl”.

2.2.3.2 Click on tab “Entities”, and then in the window “Class hierarchy” select “Thing → representation_item → manifold_solid_brep”. Check that in the window “Description” there is member “BODY_1”.

2.2.3.3 Click on tab “Entities”, and then in the window “Class hierarchy” select “Thing → representation_item → advanced_face”. Check that in the window “Description” there are members from “FACE_1” to “FACE_63”.

2.2.3.4 Click on tab “Entities”, and then in the window “Class hierarchy” select “Thing → representation_item → edge_curve”. Check that in the window “Description” there are members from “EDGE_1” to “EDGE_165”.

2.2.3.5 Quit Protégé.

3 Add Annotation

3.1 Objectives

This section is to evaluate whether OntoCAD can capture user inputs as annotations. This includes to check whether annotation anchors cover three levels of granularities, to check whether annotation are stored appropriately as OWL ontology entities, to check whether graphical user interface (GUI) is dynamically changed according to the context, e.g. different anchor and different engineering viewpoint (EV) selection cause different annotation options.

3.2 Operations

3.2.1 Anchor annotation to a body

3.2.1.1 Load OntoCAD as described in Section 2.2.1.

3.2.1.2 Click on “OntoCAD” from toolbar menu bar, and click on button “Add annotation”. Check that dialog “add OntoCAD annotation” pops up, in which there are three tabs: anchor, engineering viewpoint and annotation data.

3.2.1.3 Select the whole body of the towbar (use “selection filter” on the tool bar if necessary) and then click on button “Select” on tab “Anchor” of the dialog. Check that an item “BODY_1” appears in the list “Selected Anchors”.

3.2.1.4 Select “BODY_1” and click on next tab “Engineering viewpoint” and check that following items are listed:

- EVO_Cost
  - CostDriver
- EVO_FEA
  - Structural
  - Thermal

3.2.1.5 Select “CostDriver”, then click on button “Select”. Wait until knowledge base finishes processing (may take few seconds).

3.2.1.6 Select next tab “Annotation Data” and check that following items are listed:

- Weight
- ManufacturingProcess
- Material

3.2.1.7 Select “ManufacturingProcess” and double click. Check that dialog “Create Annotation Content” pops up, in which ManufacturingProcess and its subclass items are listed in a hierarchical style.

3.2.1.8 Scroll down the list of manufacturing processes to check if there is item “SandCasting” and select. Check that there is no “sandCasting_1” appears in the second selection box if it is not created before.

3.2.1.9 Click on expandable box. Check that a text entry box appears.

3.2.1.10 Enter “sandCasting_1” and click button “Create”. Check that a new item “sandCasting_1” appears in the second selection box.

3.2.1.11 Select “sandCasting_1” and click button “OK”. Check that dialog “Add OntoCAD annotation” reappears, in which slot “Annotation Name” is filled with “sandCasting_1” and greyed out.

3.2.1.12 Check button “Fill Data Properties” can receive focus and then click on. Check that a new dialog “Fill data for: sandCasting_1” appears, in which an item “hasLabel” exists in the selection box “Available Data Type” and Data Filler box is empty.

3.2.1.13 Select the item “hasLabel”, check that a text slot appears and no option for units.

3.2.1.14 Enter “Sand Casting” in the text slot and click on button “OK”. Check that this dialog closes and return to previous dialog “Add OntoCAD annotation”.

3.2.1.15 Click on button “Change” and check that text slot “Annotation Name” is cleared and enabled, also check that selection box “Annotation Type” is re-enabled.[17]

3.2.1.16 Select item “Weight” and double click. Check that dialog “Create Annotation Content” appears, in which an item “Weight” appears in the first selection box.

3.2.1.17 Select “Weight” and enter “weight_BODY_1_1” in the text slot to create new individual. Check that newly created individual appears in the second selection box.

3.2.1.18 Click button “OK”. Check that dialog returns to previous one, in which the selection box is locked and greyed out, so is the slot “Annotation Name”. Also check that button “Fill Data Properties” is enabled.

3.2.1.19 Click on button “Fill Data Properties” and check that dialog “Fill data for: weight_BODY_1_1” appears, in which an item “hasValue” appears in the selection box and is unselected. Check that nothing appears under “Data Filler”.

3.2.1.20 Select item “hasValue”. Check that a text slot and an option menu for units

[17] The cancellation of this operation and the following procedures are to demonstrate the dynamic interface when different annotation type is selected. The feature of dynamic GUI is also demonstrated when dealing with different anchor granularities, i.e. body, face and edge.

appear.[18]

3.2.1.21 Enter value “1.9852”, select “Kilogram” for unit, and click on button “OK”. Check that dialog returns to previous one.

3.2.1.22 Click on button “OK” and check that dialog closes.

3.2.1.23 Launch ontology modelling tool Protégé and load the ontology “C:\MechanicalEngineering\MechanicalEngineeringOntology.owl”. Click on tab “Individual” and scroll down “Class hierarchy” window to select “manifold_solid_brep” under “representation_item”. Check that “BODY_1” appears in the window “Members list”.

3.2.1.24 Select “BODY_1” and check that the corresponding “Property assertions” has a data property assertion:

- hasWeight weight_BODY_1_1

3.2.1.25 In window “Class hierarchy” scroll down to find class “Weight” under “measure_with_unit” and select. Check that “sandCasting_1” appears in the window “Members list”.

3.2.1.26 Select “sandCasting_1” and check that the corresponding “Property assertions” has a data property assertion:

- hasLabel “Sand Casting”^^string.

3.2.1.27 In window “Class hierarchy” scroll down to find class “Weight” under “measure_with_unit” and select. Check that “weight_BODY_1_1” appears in the “Members list” and select. Check that it has following object property assertions:

- hasMeasureValue mass_measure_weight_BODY_1_1_1
- hasUnit Kilogram

3.2.1.28 In window “Class hierarchy” scroll down to find class “mass_measure” under “measure_value” and select. Check that “mass_measure_weight_BODY_1_1_1” appears in the “Members list” and select. Check that it has following object property assertions:

- hasData Real_mass_measure_weight_BODY_1_1_1_1

3.2.1.29 In window “Class hierarchy” scroll down to find class “Real” under “Data” and select. Check that “Real_mass_measure_weight_BODY_1_1_1_1” appears in the “Members list” and select. Check that it has following data property assertions: hasValue 1.9852. Double click on this data property and check the value is of type “double”.

3.2.1.30 Exit Protégé.

3.2.2 Anchor annotation to a face

[18] Measure_with_unit is associated with units, which is different from manufacturing processes. This reflects the feature of dynamic GUI driven by ontologies. See Section 3.2.1.13.

This section evaluates OntoCAD anchors can be applied to faces, and demonstrates dynamic GUI.

3.2.2.1 Load OntoCAD as described in Section 2.2.1.

3.2.2.2 Click on "OntoCAD" from toolbar menu bar, and click on button "Add annotation". Check that dialog "add OntoCAD annotation" pops up, in which there are three tabs: anchor, engineering viewpoint and annotation data.

3.2.2.3 Select one face in one of the holes at the towbar base (use "selection filter" on the tool bar if necessary) and then click on button "Select" on tab "Anchor" of the dialog. Check that items "FACE_1" and "FACE_12"[19] appear in the list "Selected anchors".

3.2.2.4 Select "FACE_12" and click on next tab "Engineering viewpoint". Check that following items are listed:

- EVO_Cost
  - CostDriver
- EVO_FEA
  - Structural
  - Thermal

3.2.2.5 Select "CostDriver", then click on button "Select". Wait until knowledge base finishes processing (may take few seconds).

3.2.2.6 Select next tab "Annotation Data" and check that following items are listed:

- ManufacturingProcess
- Material

3.2.2.7 Check that item "Weight" is not in the above list.[20]

3.2.2.8 Select "ManufacturingProcess" and double click. Check that dialog "Create Annotation Content" pops up, in which ManufacturingProcess and its subclass items are listed in a hierarchical style.

3.2.2.9 Scroll down the list of manufacturing processes to check if there is item "DrillingMachining" and select. Check that there is no item appears in the second selection box if nothing is created before.

3.2.2.10 Click on expandable box. Check that a text entry box appears.

3.2.2.11 Enter "DrillingMachining_FACE_12_1" and click button "Create". Check that a new item "DrillingMachining_FACE_12_1" is updated in the second selection box.

3.2.2.12 Select "DrillingMachining_FACE_12_1" and click button "OK". Check that dialog "Add OntoCAD annotation" reappears, in which slot "Annotation Name" is filled with "DrillingMachining_FACE_12_1" and greyed out.

---

[19] The actual ID number may vary depending on the initial automatic labelling process and which whole of the towbar base is selected.

[20] This is different from anchoring body, where weight is associated with body. See Section 3.2.1.6.

3.2.2.13 Check button “Fill Data Properties” can receive focus and then click on. Check that a new dialog “Fill data for: DrillingMachining_FACE_12_1” appears, in which an item “hasLabel” exists and is unselected in the selection box “Available Data Type”, and check section “Data Filler” is empty.

3.2.2.14 Select the item “hasLabel”, check that a text slot appears without a unit option.

3.2.2.15 Enter “Twist” in the text slot and click on button “OK”. Check that this dialog closes and return to previous dialog “Add OntoCAD annotation”.

3.2.2.16 Click on button “OK” and check that dialog closes.

3.2.2.17 Launch ontology modelling tool Protégé and load the ontology “C:\MechanicalEngineering\MechanicalEngineeringOntology.owl”. Click on tab “Individual” and scroll down “Class hierarchy” window to select “advanced_face” under “representation_item”. Check that “FACE_12” appears in the window “Members list”.

3.2.2.18 Select “FACE_12” and check that the corresponding “Property assertions” has a data property assertion:

- hasManufacturingProcess DrillingMachining_FACE_12_1

3.2.2.19 In window “Class hierarchy” scroll down to find class “DrillingMachining” under “ManufacturingProcess” and select. Check that “DrillingMachining_FACE_12_1” appears in the window “Members list”.

3.2.2.20 Select “DrillingMachining_FACE_12_1” and check that the corresponding “Property assertions” has a data property assertion:

- hasLabel “Twist”^^string.

3.2.2.21 Exit Protégé.

3.2.3 Anchor annotation to an edge

This section evaluates OntoCAD anchors can be applied to faces, and demonstrates dynamic GUI. Another important point demonstrated is that new extra effort including programming is required for updating functionalities by modifying knowledge base.

3.2.3.1 Launch Protégé and load ontology.

3.2.3.2 Create a new class “EVO_Design→Note” under class “EngineeringViewpointOntology”. Fill an axiom for class “Note” as following:

- Equivalent class: hasComment some string

3.2.3.3 In the description of class “edge_curve” under “representation_item”, fill an axiom as following:

- Superclass: hasNode some Comment

3.2.3.4 Save and exit Protégé.

3.2.3.5 Load OntoCAD as described in Section 2.2.1.

3.2.3.6 Click on “OntoCAD” from toolbar menu bar, and click on button “Add annotation”. Check that dialog “add OntoCAD annotation” pops up, in which there are three tabs: anchor, engineering viewpoint and annotation data.

3.2.3.7 Select one edge on the towbar ball surface (use “selection filter” on the tool bar if necessary) and then click on button “Select” on tab “Anchor” of the dialog. Check that items “EDGE_10”[21] appear in the list “Selected anchors”.

3.2.3.8 Select “FACE_12” and click on next tab “Engineering viewpoint”. Check that following items are listed:

- EVO_Cost
  - CostDriver
- EVO_Design[22]
  - Note
- EVO_FEA
  - Structural
  - Thermal

3.2.3.9 Select “Note”, then click on button “Select”. Wait until knowledge base finishes processing (may take few seconds).

3.2.3.10 Select next tab “Annotation Data” and check that following items are listed:

- Comment

3.2.3.11 Check that no other items are in the above list.[23]

3.2.3.12 Select “Comment” and double click. Check that dialog “Create Annotation Content” pops up, in which the only item “Comment” is listed in the first selection box.

3.2.3.13 Select “Comment” and check that there is no item appears in the second selection box if nothing is created before.

3.2.3.14 Click on expandable box. Check that a text entry box appears.

3.2.3.15 Enter “Comment_EDGE_10” and click button “Create”. Check that a new item “Comment_EDGE_10” is updated in the second selection box.

3.2.3.16 Select “Comment_EDGE_10” and click button “OK”. Check that dialog “Add OntoCAD annotation” reappears, in which slot “Annotation Name” is filled with “Comment_EDGE_10” and greyed out.

3.2.3.17 Check button “Fill Data Properties” can receive focus and then click on. Check that a new dialog “Fill data for: Comment_EDGE_10” appears, in which an item “hasComment” exists and is unselected in the selection box “Available Data Type”, and check section “Data Filler” is empty.

---

[21] The actual ID number may vary depending on the initial automatic labelling process and which whole of the towbar base is selected.
[22] EVO_Design and its subclass are displayed without programmatic software upgrade.
[23] This is different from anchoring body, where weight is associated with body. See Section 3.2.1.6.

3.2.3.18 Select the item “hasComment”, check that a text slot appears without a unit option.

3.2.3.19 Enter “it is smooth line.” in the text slot and click on button “OK”. Check that this dialog closes and return to previous dialog “Add OntoCAD annotation”.

3.2.3.20 Click on button “OK” and check that dialog closes.

3.2.3.21 Launch ontology modelling tool Protégé and load the ontology “C:\MechanicalEngineering\MechanicalEngineeringOntology.owl”. Click on tab “Individual” and scroll down “Class hierarchy” window to select “edge_curve” under “representation_item”. Check that “EDGE_10” appears in the window “Members list”.

3.2.3.22 Select “EDGE_10” and check that the corresponding “Property assertions” has a data property assertion:

- hasNode Comment_EDGE_10

3.2.3.23 In window “Class hierarchy” find class “Comment” under and select. Check that “Comment_EDGE_10” appears in the window “Members list”.

3.2.3.24 Select “Comment_EDGE_10” and check that the corresponding “Property assertions” has a data property assertion:

- hasComment “it is smooth line.”^^string.

3.2.3.25 Exit Protégé.

4 Represent Annotation

4.1 Objectives

To demonstrate representing annotations previously created, it can be implied by the annotation data selection, for example the already created annotation entries for weight or manufacturing processes can be shown and selected, which implies annotation data can be dynamically retrieved and displayed to satisfy specific request according to the context, in other words, irrelevant annotation entries are filtered out to avoid distraction.

In addition, the data content of each annotation data can be displayed. In the case if the annotation is about measurement with unit, the data together with corresponding data can be retrieved and displayed; in the case if it is about literal, the text string will be retrieved and displayed.

4.2 Operations

4.2.1 Represent an free style text comment on an edge

4.2.1.1 Load OntoCAD as described in Section 2.2.1.

4.2.1.2 Click on “OntoCAD” from toolbar menu bar, and click on button “Add annotation”. Check that dialog “add OntoCAD annotation” pops up, in which there are three tabs: anchor, engineering viewpoint and annotation data.

4.2.1.3 Select one edge on the towbar ball surface (use “selection filter” on the tool bar if

necessary) and then click on button “Select” on tab “Anchor” of the dialog. Check that items “EDGE_10”[24] appear in the list “Selected anchors”.

4.2.1.4 Select “FACE_12” and click on next tab “Engineering viewpoint”. Check that following items are listed:

- EVO_Cost
  - CostDriver
- EVO_Design
  - Note
- EVO_FEA
  - Structural
  - Thermal

4.2.1.5 Select “Note”, then click on button “Select”. Wait until knowledge base finishes processing (may take few seconds).

4.2.1.6 Select next tab “Annotation Data” and check that following items are listed:

- Comment

4.2.1.7 Select “Comment” and double click. Check that dialog “Create Annotation Content” pops up, in which the only item “Comment” is listed in the first selection box.

4.2.1.8 Select “Comment” and check that there is an item “Comment_EDGE_10” appears in the second selection box.

4.2.1.9 Select “Comment_EDGE_10” and click button “OK”. Check that dialog “Add OntoCAD annotation” reappears, in which slot “Annotation Name” is filled with “Comment_EDGE_10” and greyed out.

4.2.1.10 Click on button “Display”, check that a information window pops up, in which a message represents as following:

- The data value is: [it is smooth line.,]

4.2.2 Represent an annotation of measurement with unit on a body

4.2.2.1 Load OntoCAD as described in Section 2.2.1.

4.2.2.2 Click on “OntoCAD” from toolbar menu bar, and click on button “Add annotation”. Check that dialog “add OntoCAD annotation” pops up, in which there are three tabs: anchor, engineering viewpoint and annotation data.

4.2.2.3 Select the whole body of the towbar (use “selection filter” on the tool bar if necessary) and then click on button “Select” on tab “Anchor” of the dialog. Check that an item “BODY_1” appears in the list “Selected Anchors”.

4.2.2.4 Select “BODY_1” and click on next tab “Engineering viewpoint” and check that following items are listed:

[24] The actual ID number may vary depending on the initial automatic labelling process and which whole of the towbar base is selected.

- EVO_Cost
  - CostDriver
- EVO_FEA
  - Structural
  - Thermal

4.2.2.5 Select “CostDriver”, then click on button “Select”. Wait until knowledge base finishes processing (may take few seconds).

4.2.2.6 Select next tab “Annotation Data” and check that following items are listed:

- Weight
- ManufacturingProcess
- Material

4.2.2.7 Select item “Weight” and double click. Check that dialog “Create Annotation Content” appears, in which an item “Weight” appears in the first selection box.

4.2.2.8 Select “Weight” and check that “weight_BODY_1_1” appears in the second selection box.

4.2.2.9 Select “weight_BODY_1_1” and click button “OK”. Check that dialog returns to previous one, in which the selection box is locked and greyed out, so is the slot “Annotation Name”.

4.2.2.10 Click on button “Display”, check that a information window pops up, in which a message represents as following:

- The data value is: [1.9852, Kilogram]

4.2.2.11 Click button “OK” and check that it is returned to previous dialog.

4.2.2.12 Click button “Cancel” and check that dialog “Add OntoCAD annotation” closes.

5 Evaluate Annotation Anchoring Robustness

5.1 Objectives

This section is to demonstrate the anchoring robustness, which is based on the assumption that the anchoring mechanism is robust if a NX6 part with labels can be exported into STEP file and can be imported and recognized by another CAD system – CATIA®.

5.2 Operations

5.2.1 Load OntoCAD

5.2.1.1 Save the towbar part if automatic anchor generation has been executed and export this part as STEP-203 file. Open the STEP file and check that all labels are stated.

5.2.1.2 Run application CATIA and import this STEP file. Check that all concerned geometric items are correspondingly labelled.

6 Evaluate Knowledge Sharing between EVOs and AOs

6.1 Objectives

This section is to demonstrate knowledge sharing between different engineering viewpoint ontologies (EVOs). For knowledge sharing between EVs, it can be demonstrated by using manufacturing processes in this cost estimation use case. This is based on an assumption that the manufacturing process information was automatically retrieved from a manufacturing process planning tool. Therefore this knowledge can serve the engineering cost analysis. To simulate this prerequisite, manufacturing process information is manually entered by users as in previous test procedures while adding manufacturing process "grinding" to a surface.

The operations that actually demonstrates this ability is identical to Section 7, where an application watchdog (AW) for SEER-MFG and the data exchange between OntoCAD and SEER-MFG are demonstrated.

7 Evaluate Application Watchdog (AW)

7.1 Objectives

Considering a scenario, in order to enable a cost estimation tool SEER-MFG to compute the costs for manufacturing the towbar, this requires a command file to feed in the project. A partial spreadsheet file (Table 49) containing four cost drivers is used in this evaluation.

Table 49 Example of a SEER-MFG Command Spreadsheet

Parameters	**Value (least/likely/most)**		
PRODUCT DESCRIPTION - Material	Ductile cast irons		
PRODUCT DESCRIPTION - Raw Material Cost Per Kg.	0.6325		
PRODUCT DESCRIPTION - Process	Sand casting		
PRODUCT DESCRIPTION - Finished Weight (kg)	1.9852	1.9852	1.9852

This evaluation aims to demonstrate process automation by making a judgement of an AW status through automatic reasoning over AW rules[25]. The rule for this application watchdog AW_SEER_2 defined for this scenario case is as following:

(Part or geometric_representation_item)
and (hasManufacturingProcess some PRODUCT_DESCRIPTION_-_Process)
and (hasMaterial some PRODUCT_DESCRIPTION_-_Material)
and (hasMaterialProperty some PRODUCT_DESCRIPTION_-_Raw_Material_Cost__Per_Kg.)
and (hasMeasureWithUnit some measure_with_unit)

The activity running at background during this test is reasoning over the OntoCAD knowledge base to check whether the rules are satisfied.

7.2 Operations

[25] Note that automatic processing AW is disabled in OntoCAD system to avoid unwanted processing time. This is for the purpose to save time on evaluation only. It can be re-enabled by modifying menuscript file for NX6.

7.2.1 Initially check AW_SEER_2

7.2.1.1 Load OntoCAD as described in Section 2.2.1.

7.2.1.2 Click on "OntoCAD Watchdog → Start Watchdog" from toolbar and wait until it finishes processing. Check that a dialog "Application Watchdogs" pops up.

7.2.1.3 Click on the expandable box "Functional Applications". Check the box is expanded and no "BODY_1" in the list if required data is not entered before.

7.2.2 Prepare the data model by annotating the towbar model[26]

7.2.2.1 Annotate the whole body of the towbar for material – "Ductile cast irons" by following procedures described in Section 3.2.1.

7.2.2.2 Redo procedure described in Section 7.2.1.[27]

7.2.2.3 Annotate the whole body of the towbar for raw material cost per kilogram – "0.6325" by following procedures described in Section 3.2.1.

7.2.2.4 Redo procedure described in Section 7.2.1.

7.2.2.5 Annotate the whole body of the towbar for manufacturing process – "Sand Casting" by following procedures described in Section 3.2.1.

7.2.2.6 Redo procedure described in Section 7.2.1.

7.2.2.7 Annotate the whole body of the towbar for weight – "1.9852" by following procedures described in Section 3.2.1.

7.2.3 Re-check AW_SEER_2

7.2.3.1 Load OntoCAD as described in Section 2.2.1.

7.2.3.2 Click on "OntoCAD Watchdog → Start Watchdog" from toolbar and wait until it finishes processing. Check that a dialog "Application Watchdogs" pops up.

7.2.3.3 Click on the expandable box "Functional Applications". Check the box is expanded and "BODY_1" appears in the list as following:

- Watchdog AW_SEER_2 for: BODY_2

7.2.3.4 Click button "OK" if no further tests are required, otherwise leave this dialog on.

8 Data Exchange

8.1 Objectives

This section is to evaluate that data can be exchanged between NX6 and SEER-MFG. In other words, data can be semantically retrieved from OntoCAD knowledge base and

[26] To accelerate this process, annotations can be made through modifying ontologies through using Protégé. This is desirable when testing AW rules only, instead of testing OntoCAD GUI.
[27] Section 7.2.2.2, Section 7.2.2.4 and Section 7.2.2.6 are not necessary if automatic AW is enabled.

exported into a spreadsheet as required and ready for use by an external tool SEER-MFG. This theoretically demonstrates that legacy or new applications/services can be readily incorporated into OntoCAD system by updating the knowledge base.

The semantic queries running at background during this test are reasoning activities, among which each query is executed by the reasoner and tries to retrieve the required data.

The operations are associated with previous test procedures defined in Section 7. A spreadsheet file with empty values is required as illustrated in Table 49.

8.2 Operations

8.2.1 Execute transformation agent

8.2.1.1 Double click on AW entry FOR “BODY_1” if dialog “Application Watchdogs” is on and AW entry for “BODY_1” is available. Otherwise repeat the entire or partial procedures defined in Section 7. Check that a file selection dialog pops up.

8.2.1.2 Select the prepared input file – the spreadsheet file containing required queries, and click on button “Open”.

8.2.1.3 Wait until the background process finishes. Check that an information window pops up indicating “Excel file exported for BODY_1”.

8.2.1.4 Click on button “OK”. Check that it returns to the previous dialog “Application Watchdogs”. Click on button “OK” to quit OntoCAD Watchdog function.

8.2.2 Check results for data exchange

8.2.2.1 Browse to the directory where the spreadsheet file was selected in Section 8.2.1.2. Check that a spreadsheet file named as “xxxout.xls” exists, where “xxx” represents the original name for input spreadsheet file.

8.2.2.2 Open this file and check values are filled in as illustrated in Table 49.

8.2.2.3 Close the file, exit OntoCAD, and quit NX6.